1992-09

An object-oriented approach to reliability and quality control modeling of the maintenance effort for U. S. Marine Corps Ground Combat Equipment

Mimms, Bernard F.

Monterey, California. Naval Postgraduate School

AN OBJECT-ORIENTED APPROACH TO RELIABILITY AND QUALITY CONTROL MODELING OF THE MAINTENANCE EFFORT FOR U. S. MARINE CORPS GROUND COMBAT EQUIPMENT

by

Bernard F. Mimms, Jr.
Captain, United States Marine Corps
B.S., United States Naval Academy, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1992

ABSTRACT

The U.S. Marine Corps conducts maintenance on combat essential, readiness-reportable ground combat equipment on a continuous basis. This maintenance effort is managed through a standardized database management system, known as the Marine Corps Integrated Maintenance Management System (MIMMS).

A method is developed in this thesis which provides the operational commander with an empirically based maintenance forecasting system, using information currently being collected by the MIMMS system, and producing consistently sharper local estimates of individual equipment behavior. With this method, a ground commander can specify a predetermined equipment mixture and an expected exercise duration, based on a general geographic location, and be provided estimates of equipment availability. Thus, he can better manage his maintenance effort and allocation of maintenance resources.

Forecasting is done by simulating future repair and failure times from models estimated using available maintenance history data. The simulation is configured to be managed in the MODSIM II simulation language as a series of alternating state changes, for each equipment item, up to a preselected stopping point, which would represent a projected deployment date. Estimates of equipment operational availability are computed from monitored mean failure and repair times in each state. Compilation of the prototype version, simulating six items through three complete transaction groups, is completed in approximately 15 minutes, and execution on an IBM compatible 386-25 based machine concludes in approximately 10 minutes.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PORT SECURITY CLASSIFICATION ASSIFIED	1b. RESTRICTIVE MARKINGS
CURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
CLASSIFICATION/DOWNGRADING SCHEDULE	
FORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
ME OF PERFORMING ORGANIZATION Postgraduate School — 6b. OFFICE SYMBOL OR	7a. NAME OF MONITORING ORGANIZATION
DRESS (City, State, and ZIP Code) rey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code)
ME OF FUNDING/SPONSORING GANIZATION, — 8b. OFFICE SYMBOL	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
DRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

LE (Including Security Classification)
JECT ORIENTED APPROACH TO RELIABILITY AND QUALITY CONTROL MODELING OF THE MAINTENANCE
RT FOR U.S. MARINE CORPS GROUND COMBAT EQUIPMENT

SONAL AUTHOR(S)
S, Bernard F., Jr.

E OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. Page Count
's thesis	FROM TO	1992, SEPTEMBER	143

PPLEMENTAL NOTATION
ews expressed in this thesis are those of the author and do not reflect the official policy or position of the
tment of Defense or the U.S. Government.

COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
D	GROUP	SUB-GROUP	Object-oriented, Reliability, Quality Control, Simulation Modeling, Maintenance, U.S. Marine Corps, MIMMS

TRACT (Continue on reverse if necessary and identify by block number)

e U.S. Marine Corps conducts maintenance on combat essential, readiness-reportable ground combat equipment a continuous basis. This maintenace effort is managed through a standardized database management system, own as the Marine Corps Integrated Maintenance Management System (MIMMS).

method is developed in this thesis which provides the operational commander with an empirically based intenance forecasting system, using information currently being coiiected by the MIMMS system, and producing nsistenly sharper local estimates of individual equipment behavior. With this method, a ground commander can ecify a predetermined equipment mixture and an expected exercise duration, based on a general geographic ation, and be provided estimates of equipment availability. Thus, he can better manage his maintenance effort d allocation of maintenance resources.

ecasting is done by simulation future repair and failure times from models estimated using available maintenance tory data. The simulation is configured to be managed in the MODSIM II simulation language (continued)

RIBUTION/AVAILABILTIY OF ABSTRACT CLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC	1a. REPORT SECURITY CLASSIFICATION Unclassified	
ME OF RESPONSIBLE INDIVIDUAL Whitaker	22b. TELEPHONE (Include Area Code) (408)646-3482	22c. OFFICE SYMBOL OR/Wh

BLOCK 19: Abstract (continued):

as a series of alternating state changes, for each equipment item, up to a preselected stopping point, which would represent a projected deployment date. Estimates of equipment operational availability are computed from monito mean faiure and repair times in each state. Compilation of the prototype version, simulating six items through three complete transaction groups, is completed in approximately 15 minutes, and execution on an IBM compatible 386-2 based machine concludes in approximately 10 minutes.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The author wishes to express his thanks and gratitude to the following individuals, without whom this project could not have been completed.

- My wife, Lori, for her heroic patience, understanding, and endless support of me.
- My advisor and second reader, for keeping that crack of daylight open.
- Major Dan Barnd and Captain Rudy Webbers, USMC of Marine Corps Logistics Base, Albany, Georgia, for their efforts to provide actual maintenance data.
- Captain Jim Judy, USA, for being my MODSIM programming ally, and logistics sounding board throughout this process.

Towards this goal, it is the objective of this study to develop a method which provides the operational commander from the Marine Expeditionary Unit (MEU) level through the Marine Air-Ground Task Force (MAGTF) level with the capability to monitor and analyze his own level of readiness and materiel maintenance on CEE items. Additionally, the method would provide projections of a unit's projected state of readiness, based on the unit's individual performance of maintenance actions and the reliability of the item in question, allowing individual units to determine the optimal use of maintenance resources in pursuit of maximum combat power. Ideally, this process would occur at regular intervals during non-deployment periods, but particularly prior to a major deployment, and in-theater, to manage the maintenance effort. Furthermore, the method would operate in conjunction with the MIMMS system, collecting data as it is processed, with the purpose of updating the set of distributional parameters used in the forecasting process. In this way, the method would adapt its operation to the using unit, thereby providing better estimates of the true behavior of an item of equipment.

B. PROBLEM SOLUTION APPROACH

For this study, it can be determined that on a gross scale, a system can be in one of two states: on or off. That is, ground combat equipment can be thought of as operating (on or "UP") for a time Z, and then in repair (off or "DOWN") for a time Y. Uptimes and downtimes alternate for the category code M class of items. This UP and DOWN alternating process can be modeled to determine the distribution of operational availability of the item, where operational availability over a particular time is defined

accomplished through the minimization of losses, specifically those losses that occur due to failures of equipment and the misallocation of resources.

The present method of estimating maintenance requirements bases future demand on past demand, attempts to match upcoming situations as closely as possible with similar past exercise data, and expects a limited number of excess failures of equipment. Current studies of the reliability of major weapons systems are limited due to several factors. Some major factors include the lack of specially trained analysts, the unmanageable nature of the MIMMS database, and the belief that such studies do not warrant the investiture of significant time, money or personnel towards achieving accurate analytical results.

In circumstances where no historical demands are available, such as for deployments and exercises to new geographic locations or contingency operations, maintenance requirements are determined using linear projections from existing databases of similar scenarios in conjunction with the experience of supply and maintenance personnel. These methods have proved inadequate [Ref. 2:p.27], and more often than not involve little or no examination of equipment behavioral characteristics. Therefore, an integrated system that can generate real-time readiness information responsive to the unique potential for the Marine Corps' combination of applications is greatly desired [Ref. 5:p. 1]. If this type of readiness reporting capability can be provided, it will "revolutionize the Combat Service Support (CSS) expeditionary capability of the Marine Corps" [Ref. 7:p. 2].

Furthermore, those combat essential equipment (CEE) items that fail and become incapable of performing their designed combat mission due to the need for critical repairs and have been incapable of performing the mission for a period in excess of 24 hours are considered "deadlined". They constitute a class of failures denoted by the term "category code M" equipment failures.[Ref. 1:p. 1-4] The proportion of these items of combat essential equipment that experience deadlining failures are reportable to the Joint Chiefs of Staff as indicative of the materiel readiness capability of a unit. These items are generally very expensive items, also, both in terms of procurement costs and maintenance costs. Correspondingly, this class of failures will be the defining type of principal end item failures considered.

The use of this equipment by units in operational exercises varies by unit, location, and mission type, over the span of the item's lifetime. FMF, reserve, and Maritime Prepositioned Ships (MPS) units use, maintain, and repair this equipment for exercises as varied as short-term firing exercises to months-long combined arms operations. Though the operational tempo of units is high, and the use of equipment is intense, the amount of money allocated to maintenance and repairs is limited. Additionally, by doctrine [Ref. 6:pp. 1-4], a Marine Air-Ground Task Force (MAGTF) is task organized to maximize combat power. Therefore, the number of mechanics, technicians, and other logistics personnel together with their test and repair equipment are kept to a minimum on exercises. Performance of maintenance and the management of resources are daunting and often seemingly futile tasks. Maximizing combat power, therefore, can be

Recent experiences in South West Asia and certain exercises indicate that major material readiness information is not being provided on a timely and efficient basis to the Marine Air-Ground Task Force (MAGTF) commander. This problem is further complicated by the reluctance of logistics personnel in the Fleet Marine Force (FMF) to totally rely on existing capabilities to provide maintenance related information.[Ref. 4:p. 1] The opportunity now exists to capitalize on the prototype version of the MAGTF II/LOG AIS to "design in" the capability of LFADS to manage essential information related to readiness projections and reporting [Ref. 4:p. 2]. In fact, the inclusion of these "selected items of readiness management information" is deemed essential to the Marine Corps if the overall system is to attain full capability and become the standard for FMF applications [Ref. 5:p. 1].

FMF duties are extremely diverse in terms of character, scope, length, intensity, location, and a myriad of other factors. While differences abound in these factors, similarities exist in the fact that virtually every ground combat unit uses a similar mix of ground combat equipment to accomplish their mission. Ground combat equipment tracked for readiness reporting purposes is that which Headquarters, Marine Corps regards as vital to maintaining the combat efficiency of a unit [Ref. 1:p. 1-4]. These "combat essential" items represent the fundamental materiel elements that enable a unit to conduct combat operations, and define the materiel warfighting capability of any unit. Secondary to personnel, this equipment is fundamentally necessary in pursuit of the goal of successful mission completion.

recent past performance of an individual user, scenario requirements can be forecast with greater confidence, and actual equipment use can be justified with greater reliability.

A. PROBLEM STATEMENT

The U.S. Marine Corps is in the process of employing the Marine Air-Ground Task Force Logistics Automated Information System (MAGTF II/LOG AIS) throughout the Fleet Marine Force (FMF). It has been designed to provide timely and accurate force data to the Commander in Chief of the theater involved, accurately identify lift requirements to move a force, and update the Joint Chiefs of Staff on real time force postures via JOPES (Joint Operation Planning and Execution System) [Ref. 2:p. 2]. Currently, there is a lack of accurate maintenance information reporting, and an identified inability to adequately detail the rate of the maintenance effort.

In the aftermath of Desert Storm, extensive and careful inspections of equipment employed during the war revealed much more damage than military logistics officials had expected. Equipment repairs from the Persian Gulf war are expected to cost an additional $3 billion alone, much more than initial diagnostics estimated [Ref. 2:p.27]. The current methods of estimating repairs and surveying damage is grossly inadequate, particularly in terms of forecasting equipment use under rather extreme conditions.

Within the MAGTF II/LOG AIS system, the Landing Force Asset Distribution System (LFADS) module handles the processing and management of maintenance and supply equipment. It is responsible for providing an accurate logistics posture report of the force as a whole as an operation evolves.[Ref. 3:p.24]

I. INTRODUCTION

Organizational mobility, firepower, and communication rest not only on dedication and training, but also on the ability of the supporting equipment to meet the demands placed on it. Maintenance is the logistics function of keeping that equipment properly operating. The increasing complexity, cost, vulnerability, and lethality of equipment requires an intensive maintenance effort and a corresponding respect for its employment.[Ref. 1:p.1-3] Maintenance performed on United States Marine Corps ground combat equipment is managed through the Marine Corps Integrated Maintenance Management System (MIMMS), which serves primarily as a relational database containing selected resource information on all items requiring service.

In this study, a method is developed which can ultimately provide the operational commander with the capability to forecast his maintenance requirements for major equipment end items over time, allowing him to better allocate his maintenance resources. To do this, the initial form of the life time and repair time distributions of a particular type of equipment must be estimated from past behavior of the item. The initial estimates will be based on combined data from the entire Marine Corps. Thus, they must then be revised locally and periodically updated based on maintenance information from an individual command's specific use of the equipment. Because these refined estimates take into account the general geographic location of the operating environment and the

as the amount of time an item is available for use by a unit, divided by the length of the time period. The sequence of random vectors (Z_n, Y_n), $n \geq 1$ representing a sequence of an item's UP and DOWN times will be assumed to be independent and identically distributed [Ref. 10:p. 320]. Once maintenance data is available from MIMMS, this assumption will need to be examined more closely. However, in the absence of such data, these assumptions provide a reasonable first pass at modeling equipment behavior.

There are two potential modes of employment for the model for use in the FMF. The first would involve inline placement of the model in the LFADS module of the MAGTF II/LOG AIS and would produce continuous updates of the changing readiness posture of and predictions for units preparing for exercise deployments. In this mode, the estimate of expected operational availability would give the probability that equipment would be ready for embarkation at a given time. The estimate would allow operational commanders the flexibility to reallocate maintenance assets among their units to help ensure the maximum combat effectiveness of the assigned force. The secondary mode of operation would potentially be the more prevalent employment of the model. This use would entail the day to day parallel operation of the model and the MIMMS system. By receiving MIMMS data as input transactions, the model would provide continuous updated estimates of equipment performance. At any given time, the model would provide estimates of expected operational availability to the user in a real time environment, again contributing to better allocation of resources in the maintenance effort. Additionally, it will provide an independent, individualized, empirically based analysis source for the commander regarding his own maintenance effort.

Analysis of data from the MIMMS system should provide sufficient information to identify distributional forms of failure and repair times and to compute relevant parameter estimates to begin the large-scale modeling process. Any estimates, however, can only approximate the actual individual unit's equipment behavior. Further sharpening of the parameter estimates will require more information gathered by the using unit.

To accomplish this, the approach used in this thesis is to summarize combined historical Marine Corps data in the form of a "prior" distribution for the parameters of the failure and repair time disributions. These prior distributions would be estimated using empirical Bayes estimation. The prior would then be updated at the unit level as data becomes available. After updating, the new posterior distribution of the parameter estimates give an estimate of the failure and repair distributions that are unit specific. These can then be used to forecast operational availability for the next period of time.

This can only be done by hand for relatively simple models. To be useful, implementation of this approach must allow a user to focus on the impact individual equipment behavior has on the whole group of diverse items. It must also promote real-time availability of answers, and must be easily modified for flexible scenario development.

To provide these features, forecasts of operational availability are implemented using MODSIM II. MODSIM II is an object-oriented simulation language, particularly suited to developing and operating large, process-based simulation models. Of particular importance is MODSIM's ability to dynamically allocate memory, a vital element in the successful operation of this method. The algorithms involved should pose little or no

difficulty to the local systems authority, and should provide timely and accurate solutions.

The above solution approach overcomes some of the disadvantages associated with the more classical approach currently used. For instance, by summarizing historical information of the equipment's life behavior in the form of a prior distribution, dependency on large databases of historical data is avoided. Periodic sampling and refining of the parameter estimates will provide real time revisions and permit the sharpening of forecasts. Through repeated and consistent use, operational commanders will become more confident of the model's performance and results and likewise become more adept at managing their own maintenance effort. Consistent monitoring of the model's output would indicate if the underlying assumptions are correct. Periodic re-evaluations of equipment performance would be necessary to ensure proper model operation.

C. THESIS OUTLINE

This thesis develops and presents a method for forecasting failures, repairs, and associated operational availability of U.S. Marine Corps ground combat equipment associated with the readiness reportable principal end items of a given U.S. Marine Corps FMSS supported unit. In Chapter II, the Maintenance Management Process, the information types required, an explanation of terms, the data analysis, and problems encountered during this analysis are discussed. The development of the Bayesian methodology, and justifications are addressed in Chapter III. The operation, capabilities,

and limitations of this approach are presented in Chapter IV. In Chapter V, results obtained from synthetic data, and a discussion of the empirical Bayesian technique are presented, with a delineation of computational experiences. Chapter VI contains conclusions and recommendations. Finally, listings of the General Algebraic Modeling System (GAMS) source codes for computation of initial empirical Bayesian estimators, sample I/O files, and a listing of the MODSIM II model source code are provided in the appendices.

II. THE MAINTENANCE PROCESS, DATA ISSUES, AND ANALYSIS

A primary impetus for this study was the desire to develop an empirically-based forecast model for the maintenance management of U.S. Marine Corps ground combat equipment utilizing the maintenance data resident in the Marine Corps Integrated Maintenance Management System (MIMMS). Currently, forecasts are determined by estimates calculated by knowledgeable logistics personnel or extrapolations based on historical demands. This forecasting approach fails to consider the interactive effects of several factors, mainly the type of environment, conditions for equipment use, and the individual reliability behavior of the equipment in question. It is precisely this type of operational information, that can be extracted from the MIMMS sub-files, which permits the development of a empirically-based reliability model.

A. DESCRIPTION OF THE MANAGEMENT PROCESS

Combat service support (CSS) is composed of several different functional areas and is an essential element of success on the battlefield. CSS plans, procedures, and decisions must have the flexibility to adapt to rapidly changing situations.[Ref. 12:p. 13] CSS units must incorporate and actively employ the principles of responsiveness, flexibility, and initiative, and additionally anticipate and fill the needs and requirements of their supported units before they receive requests for support [Ref. 12:p. 30].

Maintenance, and the management of the maintenance effort are primary functions of CSS, particularly in combat operations. Weapon system replacement procedures

provide combat ready assets to the combat commander. These items, which generally become unavailable due to a failure that prevents the item from performing its designed combat mission, are extremely important in the overall scope of operations, depending on the type of equipment involved. For those items of combat essential equipment studied here, weapon system replacement operations become increasingly vital towards mission success. As such, these operations are the single most difficult support concept in combat service support, due to the intense coordination among all levels of support. [Ref. 12:pp.271-4]

1. Operation of the MIMMS System

MIMMS has been established as the ground equipment maintenance program throughout the Marine Corps. It is an integrated management system encompassing all equipment commodity areas (a grouping or range of items which possess similar characteristics, have similar applications, and are susceptible to similar logistics management methods [Ref. 1:p.1-4]), based on standard policy and procedures. These policies and procedures are applicable at all levels of command and echelons of maintenance. When properly used, it contributes significantly to increased equipment readiness and causes a reduced consumption of maintenance resources. It is user-oriented and designed to work with other logistics systems, and is comprised of policies, procedures, and an information system.[Ref. 1: p.1-5]

The MIMMS Automated Information System (MIMMS AIS) provides essential maintenance management information in an efficient and timely manner. It has been designed to have the ability to provide information required to support maintenance

engineering, production, and resource management [Ref. 8: p.1-3]. The Secretary of the Navy has authorized the Marine Corps to develop a separate supply system and a separate maintenance management system to accomplish their primary missions. Both systems are designed for effective operation in both peace and war, with the capability of rapid transition from one to the other, thus making the Marine Corps essentially self-sustaining in logistics operations. Both systems are characterized by centralized management and maximum use of automated data systems.[Ref. 8:p. 1-5]

Each supported Field Maintenance Subsystem (FMSS) provides inputs on maintenance information on a daily basis. They record all maintenance activities performed by the unit on ground equipment. The procedures of the system are grouped into daily, semi-weekly, weekly, monthly, quarterly, and as requested cycles. Readiness transactions are automatically created and processed by the FMSS for all units loaded to MIMMS, based on the data submitted on the MIMMS input transactions.[Ref. 8:p. 1-9] A typical unit using FMSS assets would be a battalion sized element.

Information on maintenance actions is recorded in the maintenance shops as they occur. The FMSS records the maintenance actions by using the Equipment Repair Order (ERO). An ERO is required for all maintenance actions when maintenance resources are used by a second echelon or higher maintenance activity. Information extracted from the ERO by shop personnel is inducted into the system via the appropriate medium, usually, input to the computer. As the status of the equipment repair order changes, this information is also extracted and entered into the system. Once the ERO is closed, preselected historical information that has entered the system is transferred to

the ERO History File at the end of each month. Any or all of this information may be extracted to meet special reporting requirements.[Ref. 8: p. 2-6] Additional specific information is sent to the History Extract File, updated quarterly, which extracts files from the History File, and contains completed actions for readiness-reportable, combat essential equipment [Ref. 8:p.2-10]. This History Extract File is used to update system-wide status files maintained at Marine Corps Logistics Base, Albany, Georgia, again, on an identical quarterly basis.

The FMSS reflects the ground equipment maintenance production and the equipment readiness of selected mission/combat essential ground equipment possessed by FMF reporting units [Ref. 8:p. 2-12]. Since this is the local source of readiness information processing, it will become the focus of applications of the model.

2. Definition of Terms

Equipment designated as "combat essential" (CEE) is that equipment designated by Headquarters, Marine Corps to be vital to maintaining the combat efficiency of a unit. Included in these items are those chosen as "readiness reportable", which are those CEE items selected as representative of all equipment functional areas and whose report of status will provide the necessary data to indicate the equipment readiness of the operating forces.[Ref. 1:pp. 1-3] Equipment is described as "mission capable" if it is capable of safe use and can perform its designed primary combat function. Equipment is considered "deadlined" if it is not operationally ready; i.e., cannot perform its designed combat mission due to the need for critical repairs, and the item has been "not mission capable" in excess of 24 hours.[Ref. 1:p. 1-4]

Equipment is ultimately classified by category code on the ERO for inclusion in the MIMMS system. These category codes determine the priority of resource allocation and indicate the relative importance of the item to the unit. Category code "M" EROs are used for "readiness-reportable equipment, critical repairs which deadline the item" [Ref. 9:p. 2-2-9]. Only one category code M ERO can be opened on a specific item at any one time, at each echelon of maintenance [Ref. 9:p. 2-2-11]. Since these are the defining class of failures identifying readiness-reportable equipment for purposes of this model, and all necessary information is available, these category code M EROs will be the records of interest for analysis in this thesis.

3. Information Requirements

In order to sufficiently analyze the reliability behavior of ground combat equipment, certain key data elements must be extracted from the relevant MIMMS resource records. By examining MIMMS, a new, local user file can be created. This local user file can be created on a daily basis, to provide necessary information on readiness reportable items experiencing deadlining failures. Barring this method, access to the daily transactions is sufficient to allow necessary information collection. For each type of equipment, selected EROs provide the serial number of the item, the date of failure, the date of repair, the primary equipment operating time code (EOTC), which records the type and amount of usage experienced by the item, and the type of defect causing the failure. Table 1 provides an example of the required information.

TABLE 1
SAMPLE EXTRACT FROM LOCAL USER DATA FILE

SERIAL NUMBER	FAIL DATE	REP DATE	EOTC	DEFECT CODE
550200	90236	90240	H	A27 (TRANSPONDER FAIL)
6700013	90105	91002	R	E04 (CRACKED SEAR PIN)
550198	91102	91136	H	A17 (ANTENNA FAIL)
A123-87C	91004	91005	M	D99 (TRANSMISSION)

* EOTC = Equipment Operating Time Code
(H=Hours, M=Miles, R=Rounds, D=Days)

** Defect Codes have been annotated here for display purposes only.

To ascertain whether or not any of these factors influence the characteristic life behavior of a selected item, information should be derived from the database along several possibly related lines. By blocking the information obtained into these groups, the analysis can then focus on similarities or differences between and among the behavior characteristics of the equipment. Ideally, some usage characteristic of the item should be developed as a relevant measure of effectiveness. The number of rounds fired for weapons and the odometer mileage is occasionally recorded for major end items of equipment, but currently no formal requirement nor system for checking their accuracy exists. As a result, erroneous and oftentimes missing data limits the effective use of these indicators.

Additionally, quarterly updates of information to the Marine Corps system-wide reservoir of maintenance information prevent the accumulation of more than the last

or current 16-18 months of maintenance data. Unfortunately, the data present in this "moving window" can become subject to severe length biasing, as an item must experience "paired" maintenance actions, in order to be captured in the database. For example, an item that does not change status prior to a quarterly update, and only changes once within that quarter would not be captured in the MIMMS History Extract, which provides the data for updating; to be captured, the item's state must be defined by the corresponding "paired" transaction, i.e., closing an open record, or opening a new record. Figure 1 provides examples of captured and non-captured transactions.

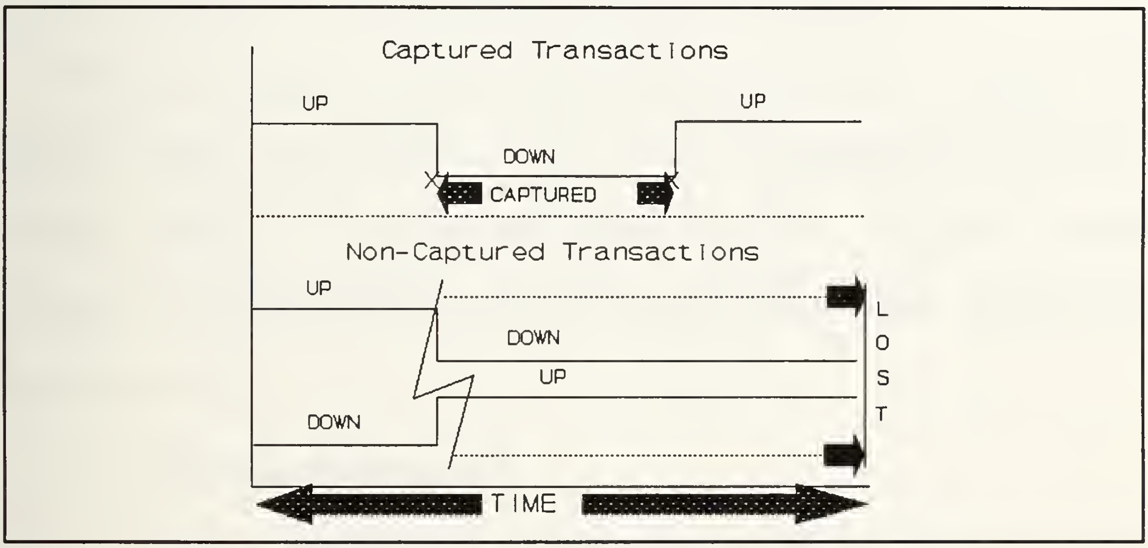

Figure 1

It should be noted that length bias sampling is still a problem when estimating the initial parameters of the from the prior distribution, however, this limitation does not affect the implementation of this version of the model. A fully capable model would require parameter estimates developed through fully monitored failure testing of equipment, to minimize the impact of length bias sampling.

Because the preferred usage data was unreliable and incomplete, the model was developed to use the dates of failure and the corresponding dates of repair to define periods of operation and repair, and subsequently, rates of respective failure and repair.

III. MODEL FORMULATION AND DESCRIPTION

In this chapter, the development of the modeling process based on an analysis of an equipment item's behavior and methods of operation, and the implementation of appropriate statistical techniques assist in describing the modeling environment are presented. The model itself is then explained in terms of its preliminary development and subsequent final phase.

A. DEVELOPMENT OF THE MODELING ENVIRONMENT

All equipment types of concern to this model experience only two major events during their lifetimes, either failures or repair. As such, the development of procedures concerning some form of alternating state process is necessary. Estimates of relevant parameters must next be computed, and some coherent policy for their updating must also be implemented.

1. Alternating State Processes

By first acknowledging the assumption that the two states, failure and repair, are independent, the procedure becomes somewhat more tenable. Intuitively, the proper functioning and repair of equipment items would seem to be a highly dependent relationship. Items that operated for extensive periods of time without failure would appear to "need" more types of repair once failure occurred. These excess repairs should correspondingly take more time. However, experience with this type of data suggests there is no correlation between the length of operating time and repair time. A possible

explanation may be in the study of the effects of preventive maintenance on the length of respective up or down times. Because of the lack of evidence to support the claim of dependence, failures and repairs are assumed to be independent in this study.

2. Estimation of Parameters and Updating

Information collected through the MIMMS system provides the necessary data for estimating failure time and repair time distributions.

a. MIMMS Transactions

MIMMS supports maintenance management by collecting, processing, and collating any maintenance or maintenance related transactions that occur to any equipment used by a unit [Ref 8:p. 2-5]. In order to do this, each equipment item's status is updated throughout the system in accordance with information input into the system by various types of coded forms of transactions, each affecting the item's status in various ways. An 0/A transaction adds a record to the transaction list. Any item requiring maintenance is added to the list through this type of transaction. An 0/C transaction changes available information on an open transaction record, should the need arise to update incorrect or newly determined information. Finally, a 9/C transaction is used to close out an open record, indicating that all maintenance actions required on this item have been completed.[Ref. 8:pp. 6-4 - 91]

TABLE 2.
EXPLANATION OF TRANSACTION TYPES

TRANSACTION TYPE	USE
0/A TRANSACTION	Initiates, or "opens" all maintenance records for use; preliminary input of maintenance to be performed
0/C TRANSACTION	Changes relevant information on any open maintenance record; record **must** be open to execute this transaction
9/C TRANSACTION	"Closes" maintenance record completely; all maintenance actions associated with opening are completed

Maintenance action on an item can be initiated with an 0/C transaction if the original reason for initiating maintenance is different from the new reason. In this way, the 0/C is used to change the appropriate required information. Similarly, a particular form of maintenance can be closed out by the 0/C, according to certain guidelines. Of importance to the model is when a transaction initiates an item's maintenance cycle, specifically readiness-reportable items experiencing deadlining failures. The corresponding exit from maintenance, or completion of all required repairs for the failure in question is the second crucial element required from MIMMS.

b. Data for Estimation

The time to failure is then determined by subtracting the appropriate return to service date from the previous failure date. The time to repair is likewise computed by deducting the repair date from the next failure date.

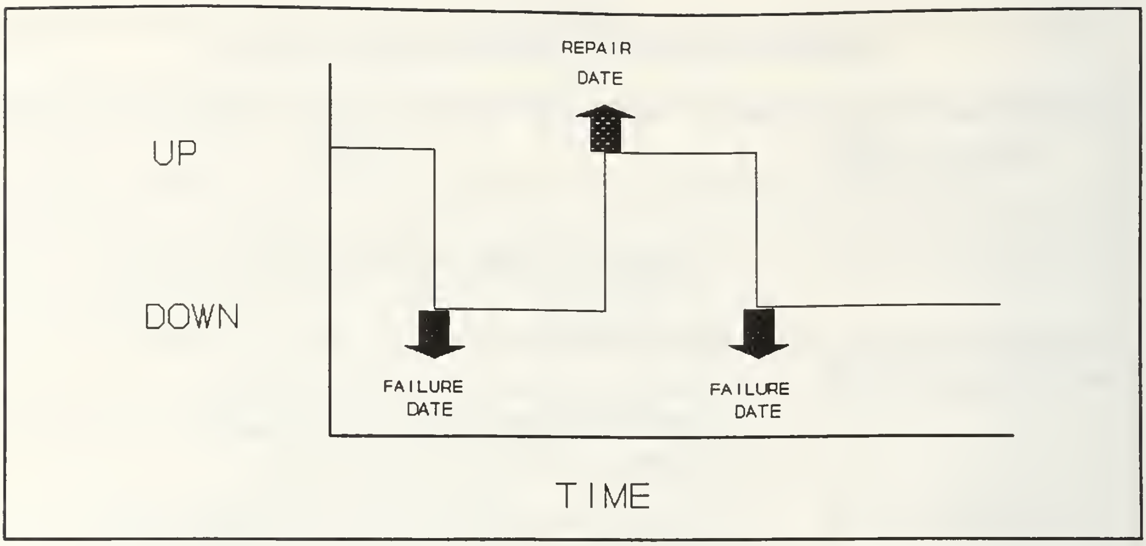

Figure 2

These dates can come from any combination of transaction types; the program must differentiate among the transaction types being input on a daily basis, associate the proper transactions with the appropriate item, and determine and perform the necessary action to maintain the item's status. Armed with these time marks and the number of occurrences, proper parameter estimation may begin.

c. *Updating of Estimates*

The original estimates of relevant distributions are determined from the aggregate data available from the maintenance information reservoir at Marine Corps Logistics Base, Albany, Georgia. This initial set of estimates must then be used in the starting versions for all units. A further enhancement would possibly be to aggregate maintenance data for each general locale, i.e., West Coast vs. East Coast, which may somewhat accelerate the process of "sharpening" the estimates at the local level.

Subsequent to the initial determination and setting of the estimates, for each using unit, individual unit usage would determine the values for use in the quarterly updating of further estimates. A reference for past empirical data, would continue to be maintained and used in the updating procedure, but this reference would eventually be overwhelmed by the more recent usage record of the individual unit. The quarterly updating period is chosen to coincide with the quarterly information transfer to Albany by all units.

B. DEVELOPMENT OF EMPIRICAL BAYESIAN TECHNIQUE

Let $[a_0, a_1]$ be a fixed time interval ($a_0 < a_1$), and let $T_{i11}, T_{i12}, \ldots, T_{i1\acute{\eta}}$ represent the sequence of up times of the i^{th} end item in that time period, where $\acute{\eta} = n_{i1}$ is the number of times the i^{th} end item is up in $[a_0, a_1]$. For example, if the item is up at the beginning of a_0 and exhibits at least one failure in the interval, then T_{i11} is the time of first failure minus a_0. If the item is up at a_1, then $T_{i1\acute{\eta}}$ is a_1 minus the time of completion of the last repair or a_0 if there have been no repairs in the interval. Note that if $\acute{\eta} = 0$, then there are no up times. Similarly, let $T_{i21}, T_{i22}, \ldots, T_{i2\acute{\eta}}$ represent the sequence of downtimes of the i^{th} end item in $[a_0, a_1]$, where $\acute{\eta} = n_{i2}$ is the number of downtimes in that interval.

Assume that for each *i* that the up times are a simple random sample from an Exponential distribution, with unknown parameter λ_{i1} and the downtimes are a simple random sample from an Exponential distribution with parameter λ_{i2}. To find the maximum likelihood estimators for λ_{ik}, the likelihood function must also account for the

fact that because sampling is over a fixed interval, these times may be censored. Left censoring at the beginning of the interval causes no difficulty, due to the lack of memory property of the Exponential distribution. However, right censoring must be accounted for. Taking into account right censoring, the likelihood of observing the sampled values $t_{ik} = (t_{ik1}, t_{ik2}, \ldots , t_{ik\hat{\eta}})$ corresponding to $T_{ik} = (T_{ik1}, \ldots , T_{ik\hat{\eta}})$, where $\hat{\eta} = n_{ik}$ for the i[th] end item, k = 1,2 and $\hat{\eta} \geq 1$ is :

$$f(t_{ik}|\lambda_{ik})=\lambda_{ik}^{m_{ik}}\exp^{-\lambda_{ik}(\sum_{j=1}^{n_{ik}} t_{ikj})}, \tag{3.1}$$

where $m_{i1} = n_{i1}$ if the item is down at time a_1 and $m_{i1} = n_{i1} - 1$ if the item is up at time a_1. Similarly, $m_{i2} = n_{i2}$ if the item is up at time a_1 and $m_{i2} = n_{i2} - 1$ if the item is down at time a_1.

This equation must be solved to get an estimate of the failure or repair rate. By maximizing the likelihood function with respect to λ_{ik}, the following estimator of the failure or repair rate is determined:

$$\hat{\lambda}_{ik}=\frac{m_{ik}}{\sum_{j=1}^{n_{ik}} t_{ikj}}=\frac{m_{ik}}{\tau_{ik}}, \tag{3.2}$$

where τ_{ik} represents the total up or down time for major end item i, for either failure $k = 1$, or repair $k = 2$.[Ref. 14:pp. 282-296]

The goal of this thesis is to develop a suitable methodology to implement an empirically based forecasting model. To do so requires some method of capturing relevant information about the model from other sources, and then being able to use it

to augment the database to get estimates of the desired parameters.

These unknown failure and repair rates can be treated as random variables. In the Bayesian framework, these parameters are each given a prior distribution, fully determined by the user, that captures his knowledge about these parameters. The data is used to update our "belief", i.e., prior density for each rate. This approach involves a Bayesian analysis of both the times between failure and the times between repair (UPTIME and DOWNTIME), to determine a posterior distribution for the failure rate and for the repair rate of a major end item. These posterior densities are referred to as the "updated" densities.[Ref. 13: p. 1]

To implement the Bayesian approach we further assume that the exact value of this parameter is a realization of a random variable with a prior distribution which is taken here to be a conjugate Gamma distribution with shape parameter α_{ik} ($\alpha_{ik} > 0$) and scale parameter β_{ik} ($\beta_{ik} > 0$):

$$f(\lambda_{ik};\alpha_{ik},\beta_{ik}) = \begin{cases} \dfrac{\beta_{ik}^{\alpha_{ik}}}{\Gamma(\alpha_{ik})}\lambda_{ik}^{\alpha_{ik}-1}\exp^{-\beta_{ik}\lambda_{ik}}, & \lambda_{ik} > 0 \\ 0 & , \ \lambda_{ik} \le 0. \end{cases} \tag{3.3}$$

The prior distribution parameters are estimated using a parametric empirical Bayes (PEB) approach for failure (and repair) rates as outlined by Gaver and Lehoczky [Ref. 15:pp. 220-224]. The PEB approach uses the entire data set to compute estimators α_{ik} and β_{ik} which are substituted into the applicable formulas. To compute the prior parameters in PEB requires the maximization of the "marginal likelihood function":

$$L_{ik}(\alpha_{ik},\beta_{ik};m_{ik},t_{ik})=\int_0^\infty(\lambda_{ik})^{m_{ik}}\exp^{-\lambda_{ik}\tau_{ik}}\frac{\beta_{ik}{}^{\alpha_{ik}}}{\Gamma(\alpha_{ik})}\lambda_{ik}{}^{\alpha_{ik}-1}\exp^{\beta_{ik}\lambda_{ik}}d\lambda_{ik}. \tag{3.4}$$

Integration results in the following marginal likelihood function for major end item i:

$$L_{ik}(\alpha_{ik},\beta_{ik}:m_{ik},t_{ik})=\frac{\Gamma(m_{ik}+\alpha_{ik})}{m_{ik}!\Gamma(\alpha_{ik})}\left(\frac{\beta_{ik}}{\tau_{ik}+\beta_{ik}}\right)^{\alpha_{ik}}\left(\frac{\tau_{ik}}{\tau_{ik}+\beta_{ik}}\right)^{m_{ik}}. \tag{3.5}$$

This marginal likelihood can then be maximized to give an estimate for the shape, α_{ik} and scale, β_{ik} parameters of the Gamma prior distribution that incorporates the actual observed operational data for each major end item.

This maximization procedure is implemented as detailed in the General Algebraic Modeling System (GAMS) model in Appendix A.

The computational variant of equation (3.5) involves the expansion of the Gamma functions in the first term. If $g(\alpha_{ik})$ is set equal to the first term, then it can be expanded about the argument along the following lines:

$$g(\alpha_{ik})=\frac{\Gamma(m_{ik}+\alpha_{ik})}{\Gamma(\alpha_{ik})}=\frac{(m_{ik}-1+\alpha_{ik})}{\Gamma(\alpha_{ik})}\Gamma(m_{ik}-1+\alpha_{ik}),$$

which, when further expanded, allows the expression of an equation without Gamma function terms:

$$g(\alpha_{ik})=\frac{1}{\Gamma(\alpha_{ik})}(m_{ik}-1+\alpha_{ik})(m_{ik}-2+\alpha_{ik})\cdots\alpha_{ik}\Gamma(\alpha_{ik}).$$

This leads to the simpler form,

$$g(\alpha_{ik})=\prod_{j=1}^{m_{ik}}(m_{ik}-j+\alpha_{ik}), \tag{3.6}$$

which is used as the computational form of the expression.

Re-computation of these estimates would need to be performed externally of the model, on a preselected time basis. Preferably, this would occur after a significantly longer time period than that assumed for the quarterly updates, as the information conferred by these alpha and beta estimates allows a greater level of resolution.

Returning to the Bayesian analysis, the n_{ik} observed times between failure (repair) and the observed exposure times ε_{ik} during which no failures (repairs) occurred are all a function of λ_{ik} and give rise to the likelihood function (3.1).

We are now interested in the updated distribution of λ_{ik} after the times between failure (repairs) have been observed, so as to provide a current estimate of future failure (repair) events. This is the <u>posterior distribution</u> and it is proportional to the product of the likelihood function (3.1) and the prior distribution (3.3):

$$f(\lambda_{ik}|t_{ik})\propto f(t_{ik}|\lambda_{ik})f(\lambda_{ik}).$$

Thus, replacing α_{ik} and β_{ik} with their estimates $\hat{\alpha}_{ik}$ and β_{ik} obtained by maximizing (3.1), an estimate of the posterior is obtained:

$$f(\lambda_{ik}|t_{ik})=c_{ik}(\lambda_{ik}{}^{m_{ik}}\exp^{-\lambda_{ik}\tau_{ik}})(\lambda_{ik}{}^{\hat{\alpha}_{ik}-1}\exp^{-\hat{\beta}_{ik}\lambda_{ik}}),$$

where c_{ik} is a constant factor whose value can easily be determined. The resulting posterior distribution is:

$$f(\lambda_{ik}|t_{ik}) = f(\lambda_{ik}|data) = \frac{(\tau_{ik}+\hat{\beta}_{ik})^{m_{ik}+\hat{\alpha}_{ik}}}{\Gamma(m_{ik}+\hat{\alpha}_{ik})}\lambda_{ik}^{m_{ik}+\hat{\alpha}_{ik}-1}\exp^{-(\tau_{ik}+\hat{\beta}_{ik})\lambda_{ik}}, \tag{3.7}$$

a Gamma distribution with shape parameter $m_{ik} + \alpha_{ik}$, and scale parameter $\tau_{ik} + \beta_{ik}$.

Since the posterior distribution of λ_{ik} has been shown to be a Gamma distribution, its estimated expected value is:

$$\hat{E}[\lambda_{ik}|data] = \frac{m_{ik}+\hat{\alpha}_{ik}}{\tau_{ik}+\hat{\beta}_{ik}}. \tag{3.8}$$

This expected value also happens to be a "Bayes estimator" of λ_{ik} when a "squared error loss function" is used. This estimator could also have been used to provide an estimate for each λ_{ik}. However, the primary purpose of the Bayesian analysis was not to provide a point estimate of λ_{ik}, but rather to provide a distribution (i.e., posterior distribution) for each λ_{ik} which could be used to make probabilistic predictions about demand functions involving λ_{ik}.

The principal reason for using Bayesian estimation procedures is that this methodology is a vehicle for admitting past experience and performance into the rate computations. By "folding in" this past information, a justified head start is developed on testing, thus less testing will be required to demonstrate required goals.[Ref. 13:pp.7-8]

The process is developed and implemented as a simulation model for several reasons. The management of the maintenance of equipment in any Marine Corps unit is a much more involved process that simply tracking maintenance information. In addition

to the maintenance of equipment, for example, modifications to and calibrations of all types of equipment must be scheduled and performed continuously.

The CEE items of concern in this thesis represent a significantly small proportion of the overall number of items that must be maintained through MIMMS. The steps necessary to accomplish the required goals of this model would require a large investment of time, effort, and personnel, all of whom must be intimately aware of CEE items' status at all times. There would be a disproportionate need to access and update several databases, which would then necessitate unplanned interruptions in the maintenance data flow. Users would also be extremely focused on the behavior of individual items, rather than the impact of item types' behavior on overall mission effectiveness.

Finally, the concept of a "simulation", where mock items are seen as theoretically performing forecasted acts in the future is more palatable to the operational commander. This is much more "believable", and saleable to most mission oriented commanders.

C. MODEL DEVELOPMENT

The process of developing the model led directly to an extensive study of the MIMMS system, and its operation as a maintenance information processor and management tool. In order to produce forecasts of the next UPTIME or DOWNTIME, for each item, as well as estimates of expected operational availability, simulation would have to be performed to some degree. Consequently, the program was named

MIMMSIM, for Marine Corps Integrated Maintenance Management System Simulation Model.

1. Process Implementation

The model initializes items, in the first step. An item type could represent a particular radio system or one type of vehicle. These items are first initialized to their starting point values, primarily, values indicating whether the item is operating (UP), or not (DOWN), and data pertaining to the item's last change in status.

The date is then referenced from that "days" transactions. Based upon a julian date computation procedure, the model determines whether the transactions are now placing demands within a new quarter or not. If not, operation continues. If the process has entered a new quarter, however, the cumulative exposure times of failure and repair, and the number of changes into each respective state, values which are accumulated within each quarter, are used to compute updated estimates of the failure and repair rates. The appropriate α and β estimates are referenced for each item, and the updating occurs as detailed previously. New estimates of failure and repair rates are now available for use within the new quarter.

Transactions, which can be input on a simulated daily basis, are examined to determine if they are of concern to the model and affect any of the items. These transactions then "update" the status of any items which they affect. From this point on, the model ceases any contact with the MIMMS system, and begins internal computations.

For any item that experiences a change of status due to that "day's" transactions, the model provides an estimate of either forecasted UP time, for items

coming out of the repair cycle, or a projection of DOWN time, for items experiencing failures.

Initiation of a "suite" simulation, one which includes all items maintained by the program, can be executed by the specification of an end date. Items are reinitialized as temporary "simulation objects", which inherit the current state and behavioral attributes of their respective prime objects. Simulation alternatingly switches between the UP and DOWN states, to the selected stopping date, tracking and accumulating the number of occurrences and exposure time information on each item type's respective objects.

2. Computation of Output

For each day's transactions, any item that changes state due to that day's transactions is provided an estimate of its expected UP or DOWN time based on a call to an Exponential distribution with the current lambda rate parameter estimate for that item type.

The suite simulation monitors the amount of time each item spends either UP or DOWN; by computing the average of each of these values, the expected operational availability can be estimated as follows:

$$A_o = \frac{AVG[UP]}{t} \tag{3.9}$$

where AVG[UP] is the average uptime and t represents the downtime over all simulation repetitions. For this model, the simulation is repeated 500 times, to better determine the average state times.

This result is important because it is based on the cumulative data for each item run through the simulation. The use of these empirically based estimators to ascertain the projected operational availability significantly expands the possible realization of a more accurate assessment of equipment reliability. These forecasts are much more likely to be accurate than any available, through current methods. In addition, the distribution of operational availability is estimated by the histogram of the availabilities from the 500 repetitions of the simulation runs.

IV. CAPABILITIES AND LIMITATIONS OF THE MODEL

The specific model developed for the demonstrative purposes of the thesis demonstrates the potential of a more general model, which would be employed for actual use. To illustrate the operation of this proposed general model, the specific model considered here performs all of the functions of the planned larger model, using synthetic items of equipment and transactions. By demonstrating the power of this methodology on a smaller scale, maintenance personnel will better appreciate its operation when implemented at their level of use.

A. SPECIFIC OPERATION OF PROTOTYPE MODEL

The prototype model, whose source code is provided in Appendix D, was written in MODSIM II, an object-oriented simulation modeling language, which allows the development and implementation of large, process based simulations. Traditional languages, such as FORTRAN, do not allow the allocation of memory on a dynamic basis, a function which would enable the model to handle different volumes of transaction traffic, on a daily basis. Towards this end, and because of the inherent flexibility and power of MODSIM II, the model was developed in this language in such a way as to allow easier expansion when full scale development ensues.

The model, provisionally entitled MIMMSIM, for Marine Corps Integrated Maintenance Management Simulation Model, develops and maintains maintenance information on two different types of items (separate ID numbers), each with three

individually serialized items each, for a total of six items. These items would represent readiness-reportable items in a unit. Three "days" worth of transactions are sequentially processed against the items, updating each item's status as necessary, and also determining whether the current date will initiate an update of the relevant parameter estimates, based on the methodology developed in the previous chapter, and using the cumulative state values computed and maintained throughout model operation.

Each "day", the first transaction's date determines if a required update is necessary. After revising the status of any items affected by the transactions processed, the program references each item's appropriate estimate of either failure or repair rate, based on the most recent quarterly update. Each item that has changed state due to that "days" transactions is then provided with an estimate of its expected "life" with its new state. These forecasts are based on the most current empirically determined estimators of true equipment behavior.

Before continuing to the next "day's" transactions, the program asks the user if a suite simulation is required. The entire "suite" of items managed by the program (which would include all readiness-reportable items in the full model) would be prepared for simulation should the user answer "yes". Preparation of the items involves a determination of each item's current state, either UP or DOWN. The user is next prompted for a stopping date, which is translated to simulation time by the program. This stopping date would most likely represent some date of real or potential action for the unit and its equipment in the future. Most stopping dates for the model, used in its

desired placement, the LFADS module of the MAGTF II/LOG AIS system, would be projected deployment dates for the monitored unit.

The simulation is initiated, and items are run through the system, switching between their current state and their alternate state, up to the preselected stopping date. This process is repeated 500 times for the model, with different segments of random streams used each iteration. The program monitors the times of both UP and DOWN states for each item, compiling each for use as aggregate ID number type mean estimates. Specifically, this prototype produces an estimate of average UP and DOWN times for each item type. The operational availability is finally computed for each type of object, based on equation (3.9).

This operational availability estimate provides the operational commander with a percentage of operable items expected to be available on the preselected stopping date, based on an empirically based estimate of equipment behavior and the results of a process based simulation.

The model continues on, processing the next "day's" transactions, performing essentially the same functions as before. By expanding the same model, adding more items and several more days worth of transactions, a user can gain even more of a sense of operating the model in a real-time maintenance environment. The modular nature of MODSIM II allows the realization of these steps to be a simple process, mainly time consuming in the implementation, rather than the development of new code.

B. OPERATION WITHIN THE MIMMS ENVIRONMENT

For expansion of the model for use within the MIMMS arena, the operating interface need only have access to the daily transactions processed through MIMMS. The initial parameter estimates, to start each version of the model, are derived from the information provided to MCLB Albany, Georgia, through the MIMMS History Extract File. This 16-18 month mass of maintenance should provide sufficient empirical weight to this particular version of the model. The daily transactions can best be obtained from the Daily Transaction Listing (DTL), which collects all transactions input through the system on a daily basis [Ref 8:pp. 17-9 - 11].

After receiving the daily input, the model ceases contact with MIMMS. All maintenance of state times and occurrences takes place within MIMMSIM, thereby minimizing the possibility of data corruption in either direction. MIMMSIM should and does operate with as little interference as possible to the MIMMS system.

Given these transactions, the process should continue along the lines detailed for the smaller, prototype version of the model. The only difference is in the quantity of information being processed, not in the methodology being employed. The simple methods employed in the prototype aid in the expansion to the fully implemented model, by allowing simple, streamlined, and logical operation in its own simulation environment.

C. CAPABILITIES USING AVAILABLE MIMMS INFORMATION

There are many advantages in using the available maintenance information from MCLB Albany, Georgia to compute starting point parameter estimates for the versions

of the model initially placed for inline use. Since failures and repairs can be affected by the location in which they occur, due to the location of supply sources, weather, usage, etc., those locations could have significantly different realizations of parameter estimates than those of similar equipment from another location. By sorting the system-wide database by locations, the available data can provide an even better "starting point" estimator. Projections would be sharper, and users would place more trust in their system MIMMSIM models.

Additionally, the specific location parameters could serve as some form of "baseline" estimators for the general geographic area. Marine Corps units operate from five basic activity codes around the world. Each activity code is roughly comparable to the general geographic area of concern. The major areas of maintenance concentration are the east and west coasts of the United States, Okinawa, Japan, the Reserve units, and Hawaii [Ref. 16: p. 1]. Assuming that these estimates are sufficiently different, units preparing to deploy to the general geographic area of a MIMMS supported unit can benefit from liaison with those units.

The estimators used in the model could also serve as quality control monitors, indicating the validity of certain techniques or materials used in the maintenance effort. The primary operation of this model would provide increasingly better, or "sharper" estimates of the relevant parameters describing the reliability behavior of equipment used by Marine Corps units.

D. MODEL LIMITATIONS

The model is limited in its use only by the data provided from equipment usage to determine parameter estimates. The data available for use is extremely noisy. Various mistakes in the individual unit's data input process and artifacts of the data itself make even a limited analysis of the available data challenging. Equipment usage is better described through some measure of actual *use* of the item rather than the determination of the number of days since its last repair. At the present time, however, the Equipment Operating Time Code (EOTC) is not a mandatory entry for maintenance reports. Even so, it is rarely reported or recorded consistently and accurately. For definitive descriptions of equipment reliability behavior, further study in this area is necessary.

The model cannot otherwise begin to accurately detail the true behavior of modeled equipment. Further study in this area would undoubtedly involve analysis of various elements of the maintenance process, leading to a greater understanding of the underlying mechanisms and stronger and more precise estimates for use in the future. In particular, repair times are more likely to have a distribution with heavier tails than an exponential, and failure times should exhibit increasing or bathtub shaped failure rates. In these cases the empirical Bayes approach can still be used for more realistic models, but computational tractability will be lost, and estimates will have to be found numerically.

V. SAMPLE IMPLEMENTATION OF MIMMSIM

The results detail the impact of the prototype model. Comparison of model results to actual maintenance forecasts is not possible, as there exists no systematic, coherent, empirically based maintenance projection capability other than MIMMSIM. The results in this section are only intended to illustrate the potential of MIMMSIM. Further analysis of the maintenance process is required to fully define the distributional behavior of equipment items. Given this information MIMMSIM is fully capable of operating as described, and providing all output detailed previously, with a minimum of additional coding.

A. MODEL TESTING

The data used for the execution of the MIMMSIM Prototype model was completely synthetic. In developing the data, similarity with actual data was desired. Two item types, A1000 and B2000, were used in the process. Serial numbers of individual items were assigned in order, i.e., A1, A2, etc., by item type. By sequentially adjusting the rates used in the model, the program eventually received two sets of parameter estimates for each item type (a set for failures and a set for repairs). The real initial parameter estimates would be computed through use of a program similar to the GAMS program located in Appendix A, which utilizes equation (3.6), the computational variant of equation (3.5).

With three synthetic "days" worth of transactions processed through the model, the model provided six estimates of projected operating or repair time, three of each type. Each day, a complete "suite" simulation was performed to ascertain if any changes occurred due to maintenance activities. Each simulation was performed for 60, 150, and 300 days, to determine system wide simulation behavior. Each "suite" simulation was repeated 500 times, with the average up and down times compiled for all items through all runs. The simulation was performed using different random number streams for each item, with subsequent runs accomplished through realizations of each stream. The 500 runs produced a better estimate of expected operational availability of each item than a single run. In addition, the 500 runs provide an estimate of the distribution of operational availability. An increase in the number of runs past 500 proportionally increases program run time, with very little increase in the efficiency of the estimate of estimated operational availability.

The process moved to a new quarter with the second "day's" transaction, to illustrate the parameter estimate updating procedure. Day three included additional transactions further detailing the behavior of the items. Table 3 summarizes the results for the complete operation of the model, under these conditions.

TABLE 3.
SUMMARY OF RESULTS FOR PROTOTYPE OPERATION SCHEME

	Day 1	Day 2	Day 3
Daily Projections (Days)			
	A2: 2.33 UP	A1: 6.07 DOWN	A2: 2.85 UP
	B2: 1.07 DOWN	A2: 7.34 DOWN	A2: 15.08 UP
Average Operational Availability (%)			
60 Days	A1000: 65.828 B2000: 83.775	A1000: 62.920 B2000: 84.197	A1000:62.045 B2000:84.023
150 Days	A1000: 67.096 B2000: 87.457	A1000: 66.384 B2000: 87.704	A1000:63.521 B2000:87.686
300 Days	A1000: 67.656 B2000: 88.575	A1000: 67.350 B2000: 88.672	A1000:64.292 B2000:88.678

1. Estimation Performance Testing

To illustrate the importance of simulating to get estimates of expected operational availability over a relatively short time period, in this section operational availability is forecast for 10, 20, 30, 40, 50, and 60 days for items A1000 and B2000. The box plots of Figures 3 and 4 indicate the variable nature of the forecasted operational availability, for each item type. Using renewal theory, it can be shown that expected availability approaches the ratio of expected up times to the sum of expected up times

and down times as the time period increases to infinity. Thus, for long periods of time, expected operational availability for item i can be estimated by

$$E(A_o) = \frac{\hat{\lambda}_{i1}}{(\hat{\lambda}_{i1} + \hat{\lambda}_{i2})}.$$

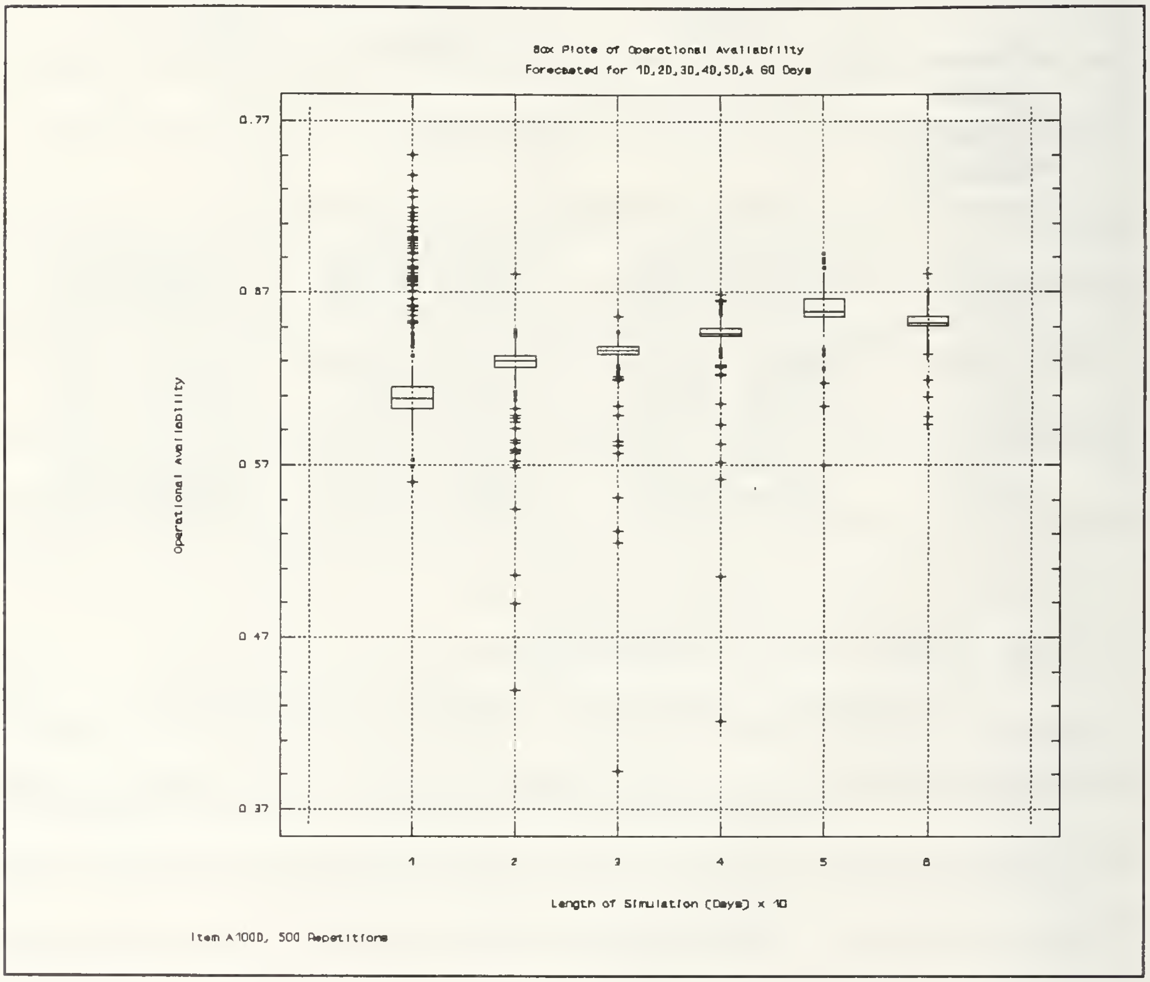

Figure 3

For item A1000 and B2000, this gives estimated operational availability estimates of 68.08% and 89.475%, respectively. From Figure 3, it is clear that even at

40 days, the average operational availability for item A1000 is close to the estimated limiting value of expected operational availability. However, for item B2000, the average operational availability is not even close to 89.475% at 60 days.

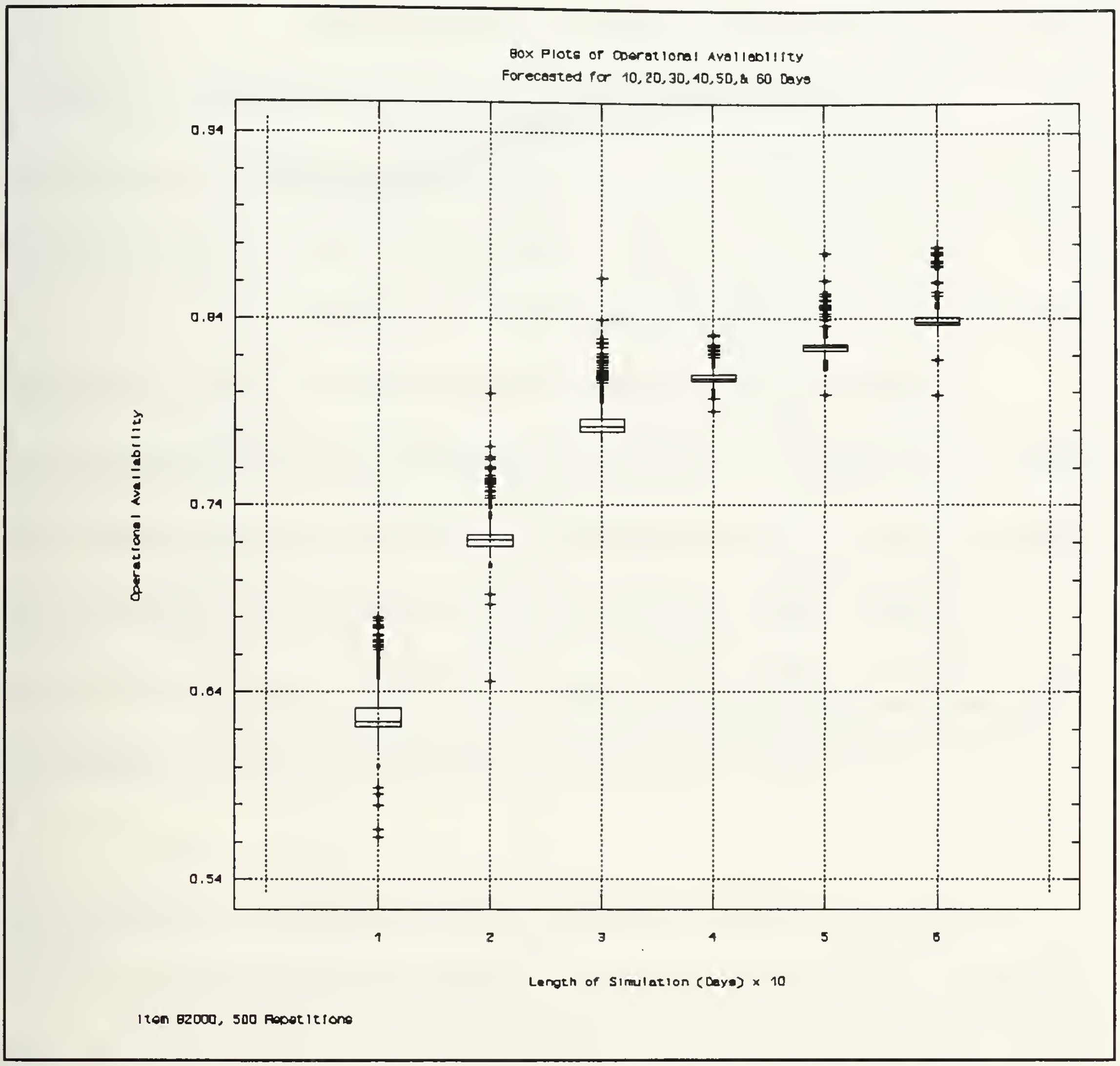

Figure 4

Since the model only simulates three items of each "type", the starting point status of items will heavily bias the operational availability of the shorter term simulation runs. An item with an estimated "UP" time of 17 days and estimated "DOWN" time of

13 days, for instance, will display lower availability for a simulation run of length ten days if two of the three items begin in the "DOWN" status position, and correspondingly higher availability if more start out "UP". Over a longer simulation period, however, the system has an opportunity to "smooth out" early status effects.

It is also clear from figures 3 and 4 that average operational availability does not give a completely accurate picture of what the operational ground commander should expect. The variability and the lack of symmetry of the distribution of operational availability is even more important than the average availability.

Another way of approaching the interpretation of availability is to forecast the number of "UP" items on any given day. This affords the user the opportunity to bracket segments of time where potential availability would be lower than acceptable limits.

Figures 5 and 6 attempt to illustrate this concept by providing the average number of "UP" or available items averaged over the three items of type A1000, and the three items of type B2000, respectively, for each day of several different length simulation runs.

Item A1000 displays steadily increasing "availability" with time, after an initial drop. For this particular setup of the model, item A1000 had all three of its items begin the simulation terms in the "UP" status. Due to this condition, the separate simulation runs are much more coherent in their patterns, as well as consistent in individual daily trends. The items tend to alternate between up and down states together throughout simulation runs, resulting in reasonably consistent results.

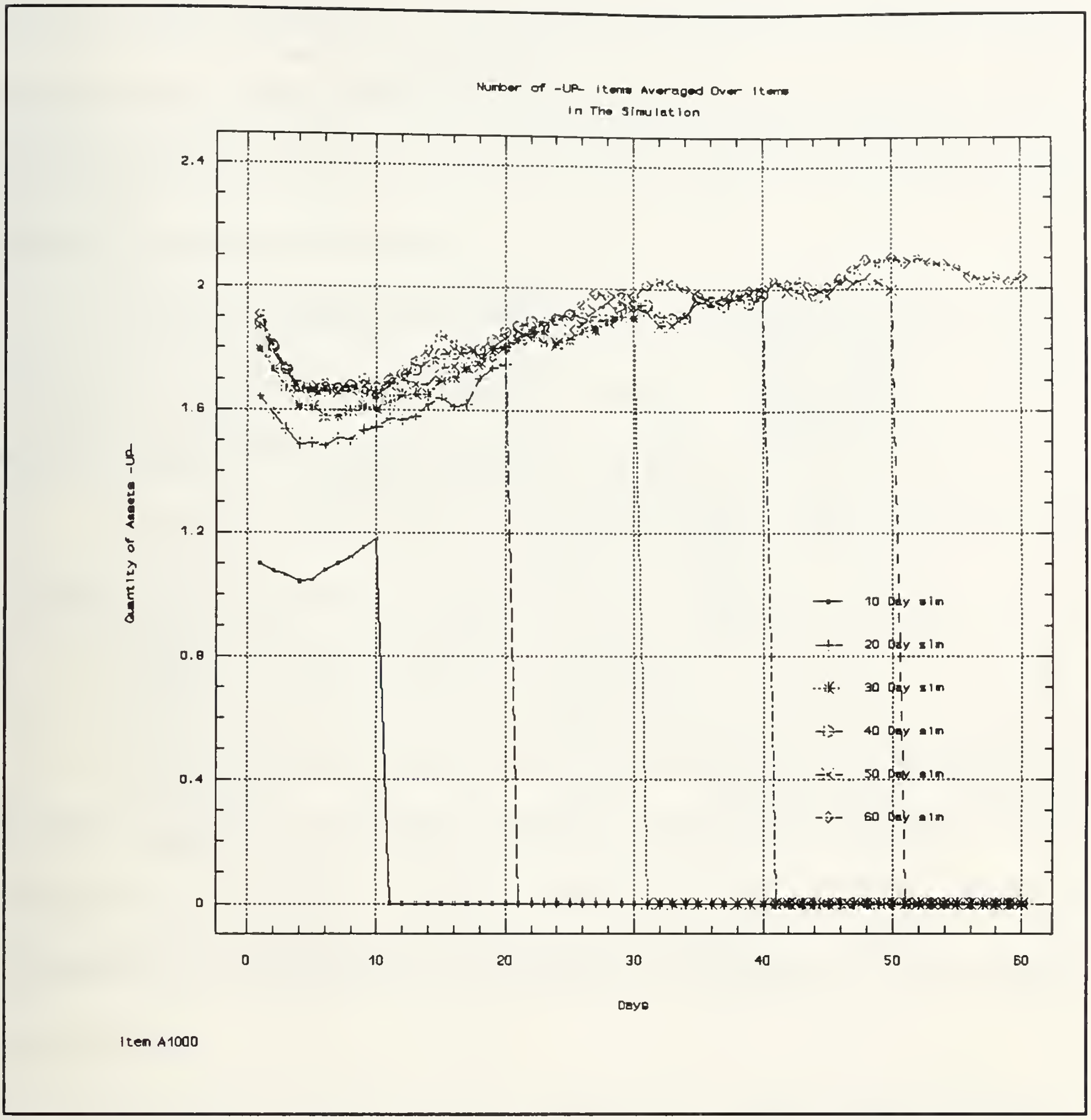

Figure 5

The average number of items available correspond to operational availability estimates in the sense that these quantities represent the available numbers of items for use through time, where the operational availability is an estimate based upon the aggregation over time of information from this continuing alternating process.

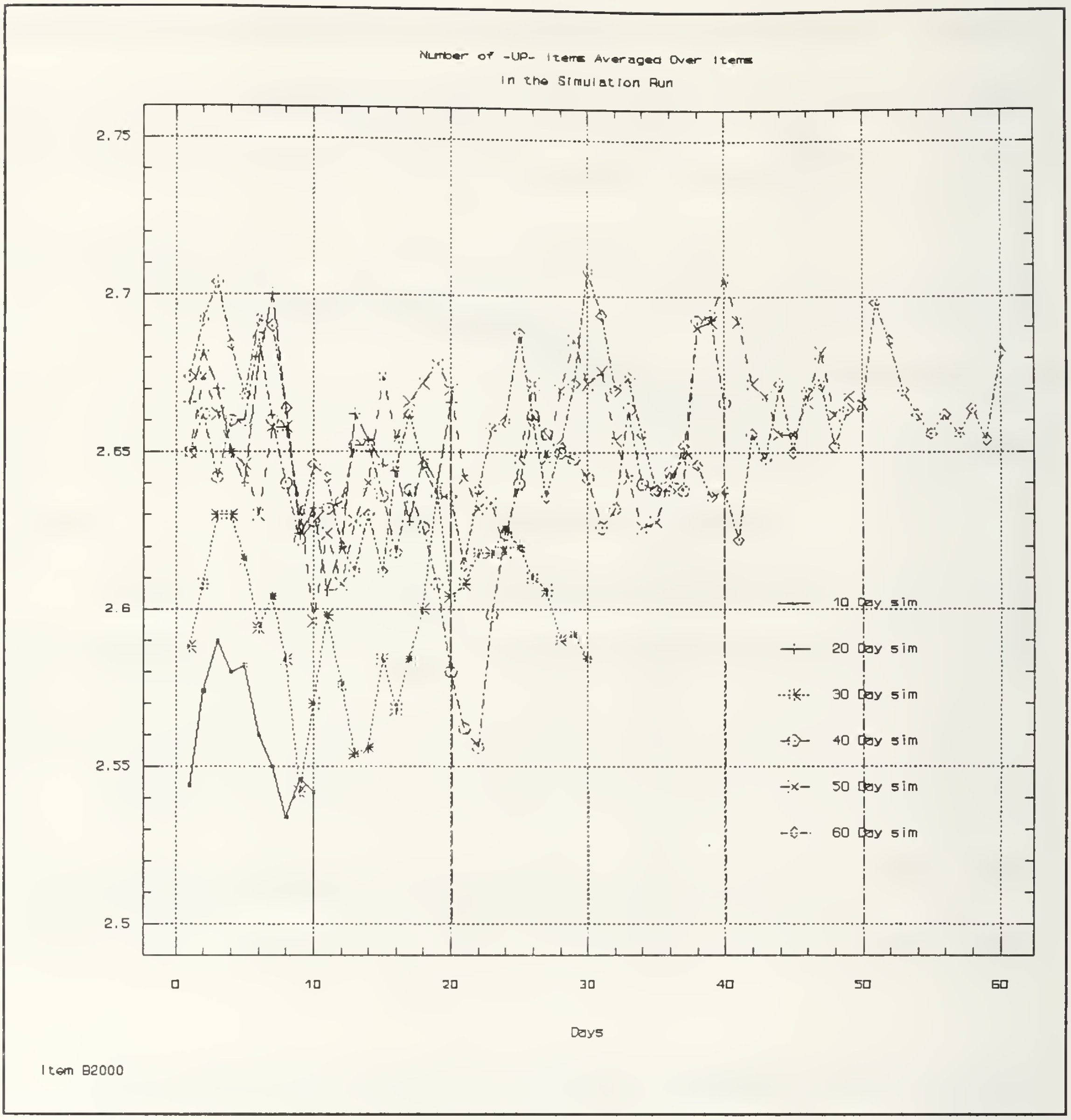

Figure 6

Item B2000 is much more varied in its determination of the quantity of available items. This item does not display the same behavior as item A1000 in the cyclic nature of "availability".

The answer could lie in the starting point status of each item type. Again, the results are highly dependent on individual equipment behavior, due to the small number

of items considered, a predominance of either "UP" or "DOWN" initial statuses for any item type could have a large influence on final results. In this case, the A1000 type items all began simulation in the "UP" state. The strong correlation between the quantity available vs. time plots confirms this.

The B2000 items begin simulation with two items "DOWN" and one "UP", in this case, a much more diverse mix. This initial state would explain the greater variability evidenced in the early time periods.

As a rough indicator of operational availability, the quantity of items "UP" is a measure of effectiveness that needs to be explored further.

2. Comparison of the Model as an Estimator

The model was developed as an availability predictor using the exponential distribution. In reality, the underlying distributions of the failure and repair times of equipment items will likely be different. The model must be flexible enough to accommodate the implementation of supplementary distributions, and still produce accurate estimates.

The model was thus tested and compared to several different schemes, to evaluate its performance in the presence of alternative equipment behavior. The Gamma distribution was selected because of its flexibility, and applicability to the modeling of various equipment operating and repair times.

To accomplish these comparisons, one item of A1000 type was run through the model for the various schemes, starting in the UP state, tracking its availability status for each. The appropriate schemes are detailed in Table 5.

TABLE 5.
COMPARISON SCHEMES

SCHEME/PARAMETERS	λ_U	λ_D
E	Exp(14.57)	Exp(6.83)
G2	Exp(14.57)	Gamma(2,3.415)
G2A	Gamma(2,7.285)	Gamma(2,3.415)
EX (Updated Params)	Exp(6.1998)	Exp(5.046)

The EX scheme was run for 20 days, then the failure data generated was used to provide information to update the up and down parameters for the completion of a 60 day run. This was done for comparison purposes only, and is not an indication of actual working of the model.

The two Gamma schemes were chosen to illustrate an increasing rate of either failures or repairs, respectively. The comparison of the strict exponential case as a forecaster can be evaluated against these possibly more realistic schemes.

Figure 7 details the performance of the model using Scheme E, for 20 and 60 day forecasts.

Starting in the UP position, we would expect the 20 day simulation to provide higher estimates, simply because the mean UP time is so near this 20 day cutoff period. The 60 day forecasts, in contrast, are slightly lower, and closer to the target availability value because the model has had sufficient time to "settle down" to this value. These results are not contradictory to those shown in Figure 3, which involved the modeling

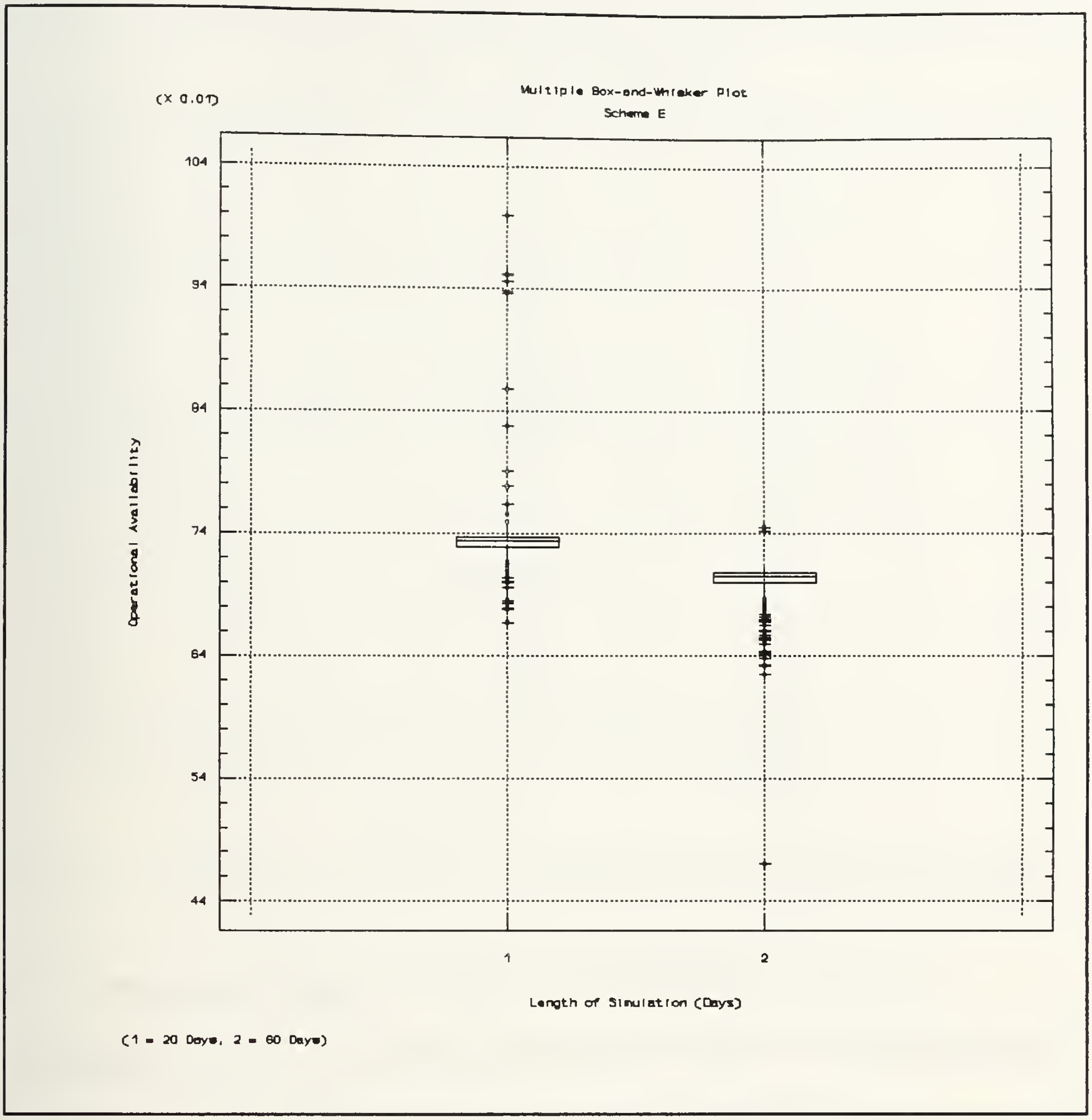

Figure 7

of three individual items through time, and the computation of availability as a function of the proportion of items available for use.

Figure 8 details the performance of the model using Scheme G2, for 20 and 60 day forecasts.

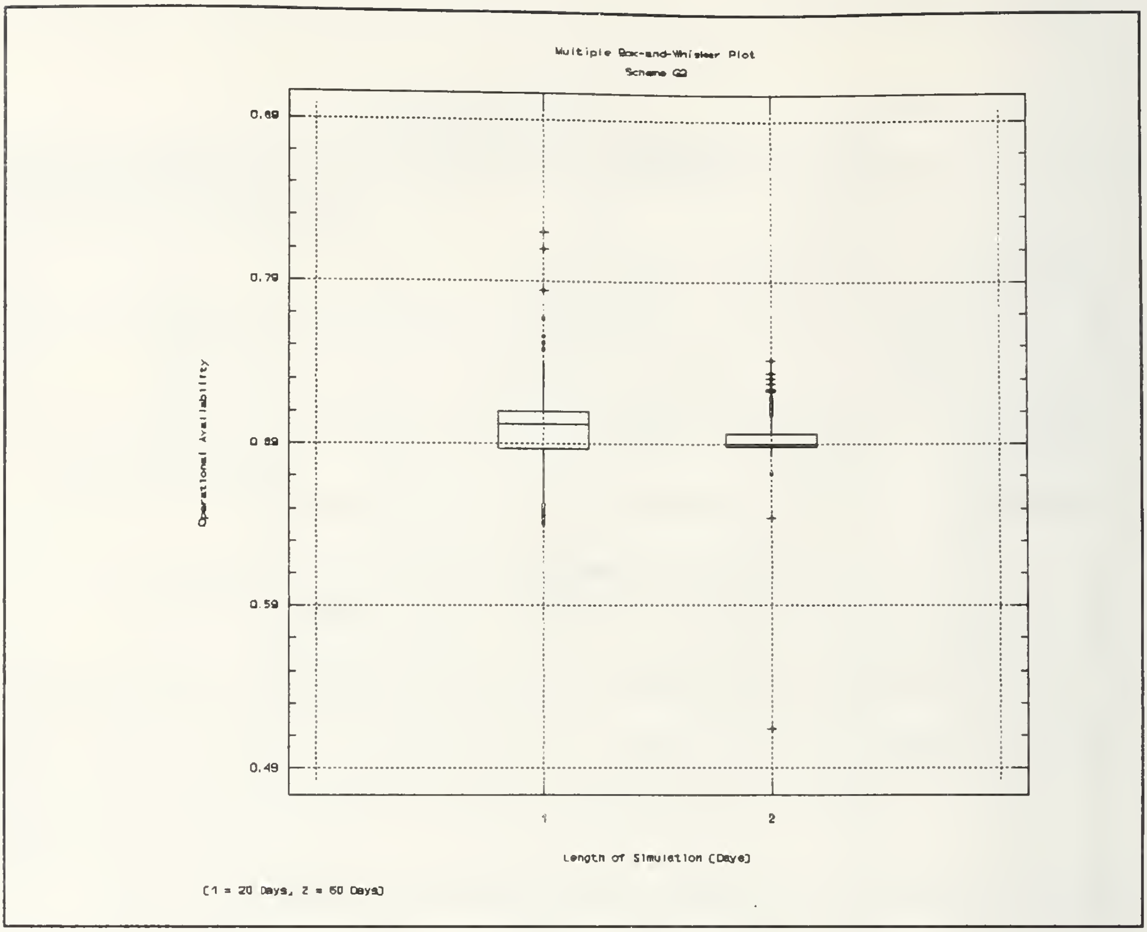

Figure 8

In this case, the repair times are effectively shorter, thus the item should be "UP" or available more. The same limitations appropriate for scheme E apply as well, with slightly higher availability expected at 20 days versus 60.

Figure 9 details the performance of the model using Scheme G2A, for 20 and 60 day forecasts.

The availability is much higher in this case. The parameters chosen provide decreasing times of both failures and repairs. The operational availability is thus much greater in the short run, and still relatively high at the 60 day point. Figures 10 and 11

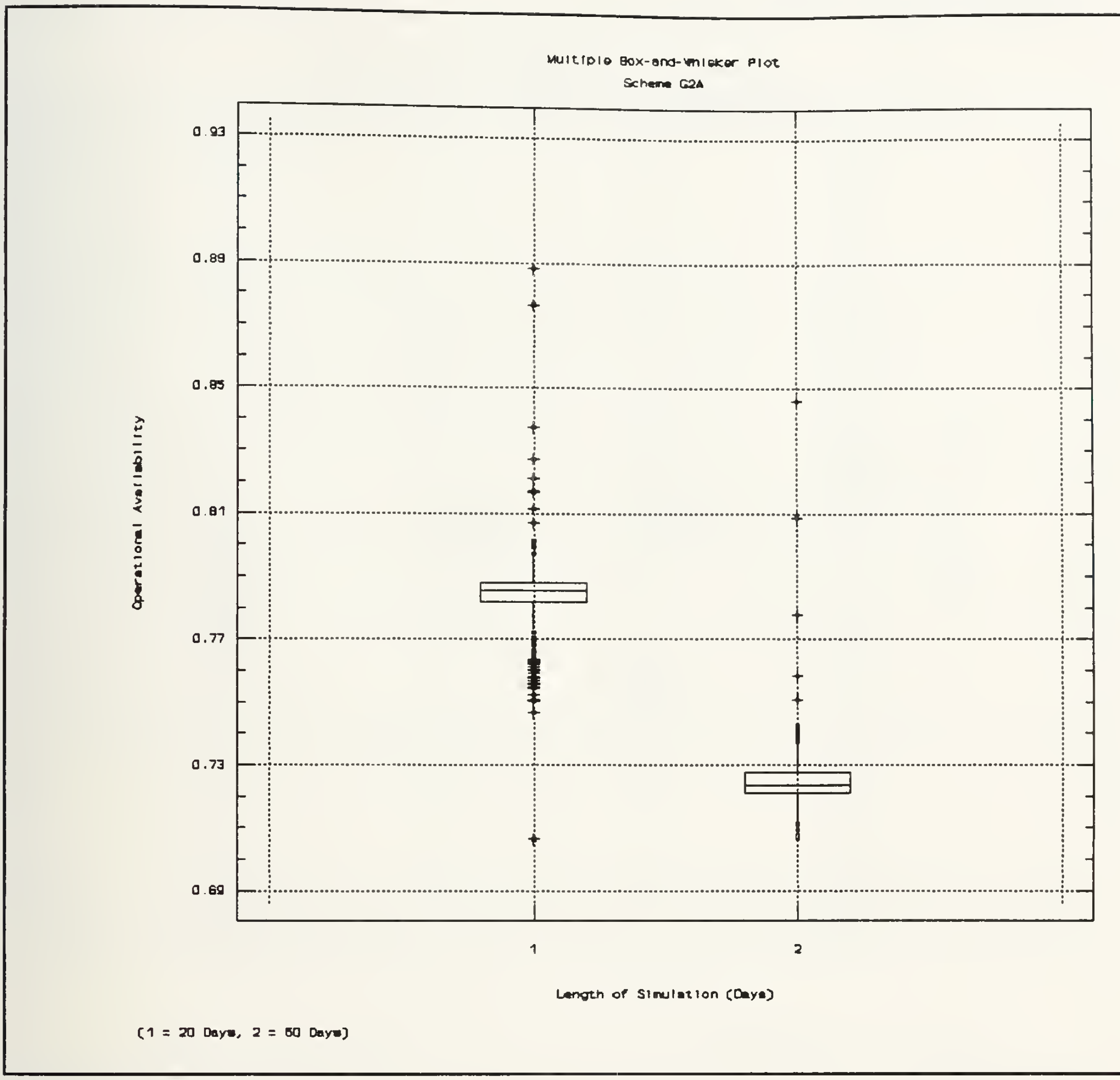

Figure 9

outline the comparisons of each scheme's forecasts for 20 and 60 days, respectively.

Scheme G2 actually provides the most conservative estimate of operational availability. In comparison, the strict exponential case comes within approximately 4% of scheme G2's value, and approximately 5% of scheme G2A.

At 60 days, the predictions are much closer, scheme E approximating within 0.6% of scheme G2, and 2.6% of scheme G2A.

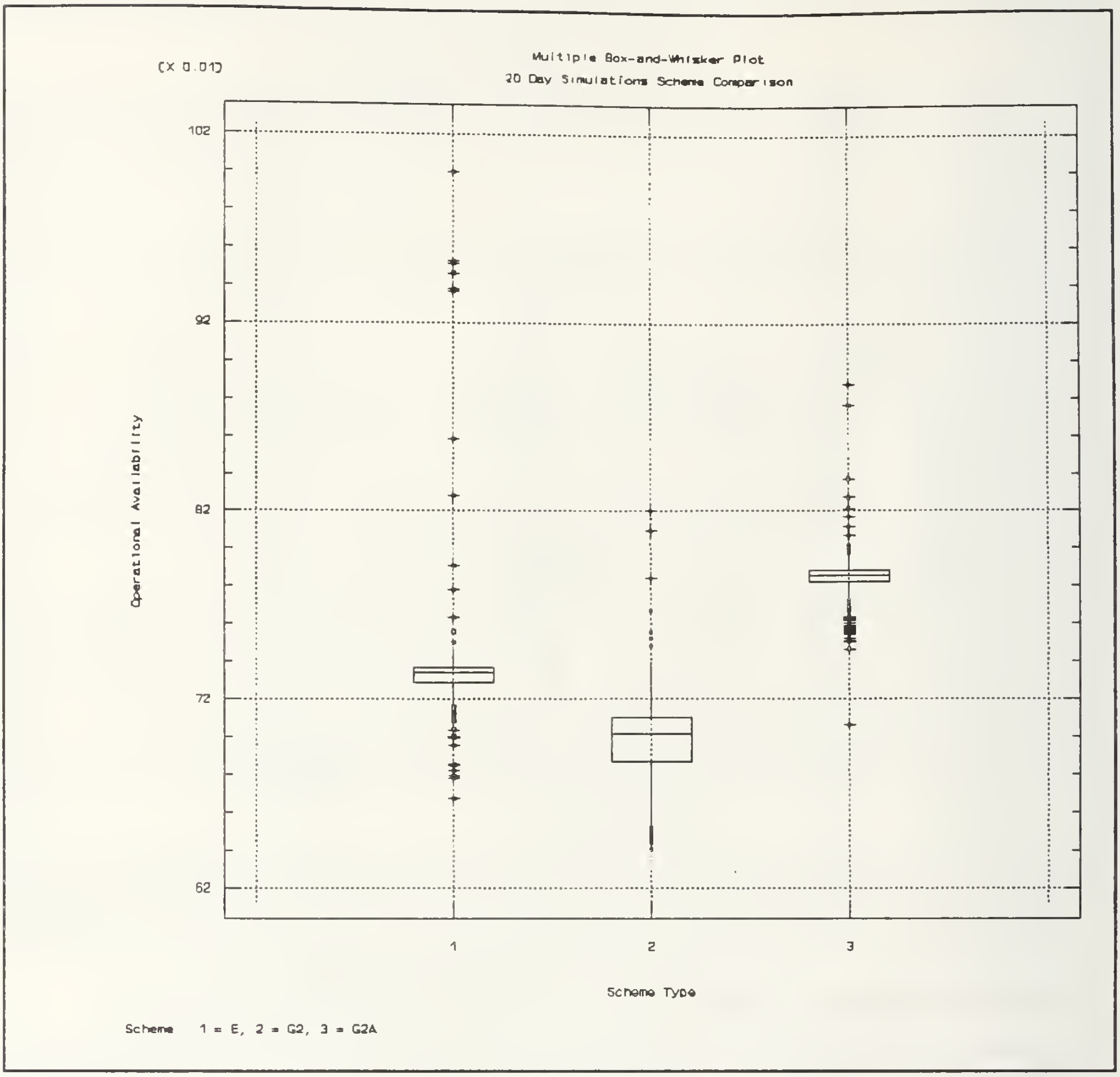

Figure 10

The true test, however, is in comparison to the model as it updates the parameters used to compute forecasts. Scheme EX was developed to produce 20 days of simulated failure data, which was then used as synthetic data to update the model's failure and repair parameters used in the exponential case, for the completion of the 60 day simulation. This is to approximate the effect of updating the parameters after actual usage, which the model was designed to accommodate. Figure 12 shows the comparison

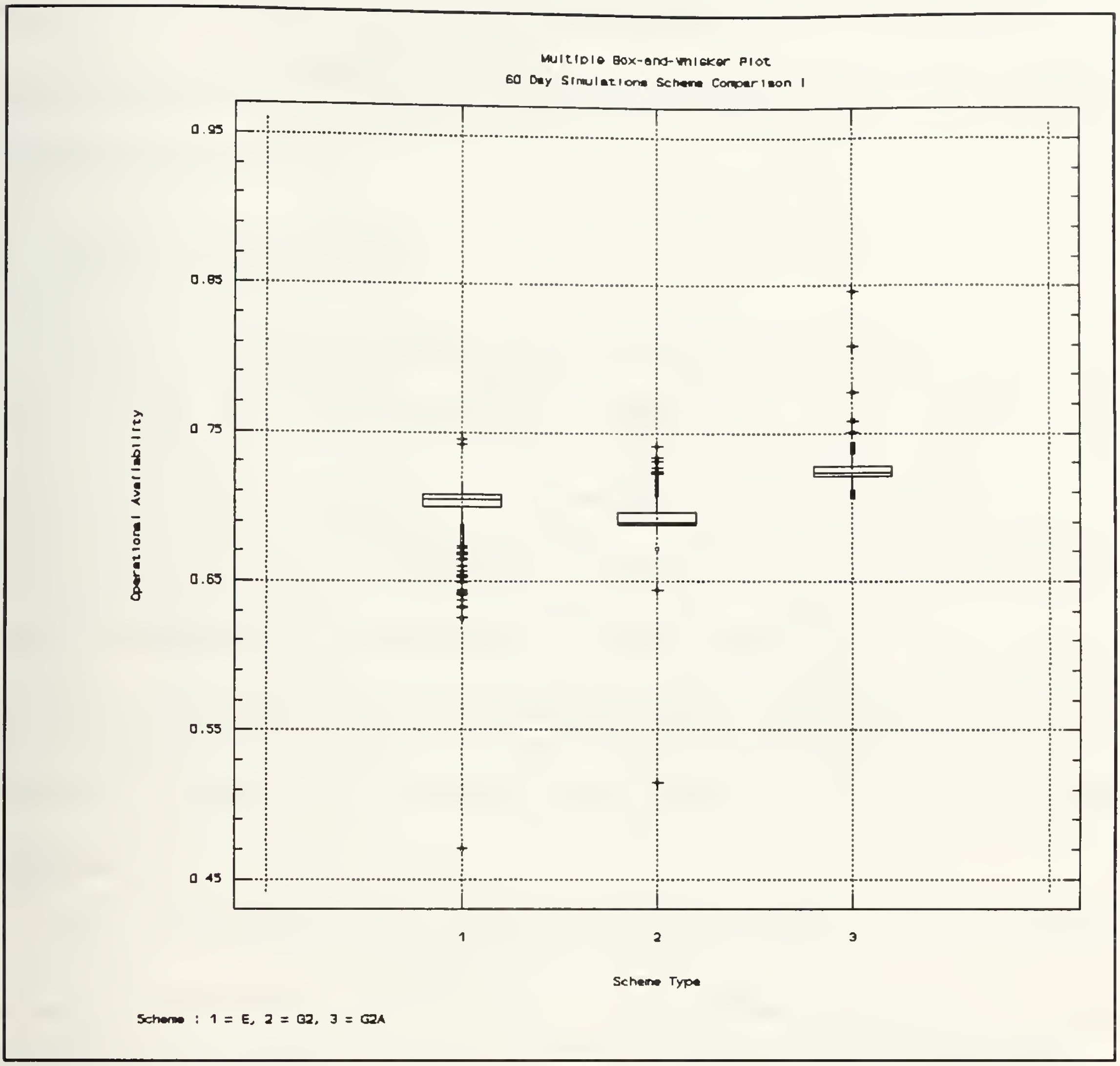

Figure 11

of predictors including the scheme EX case.

The obvious effect of Bayesian updating through the exponential case is to keep the forecasts from exceeding the "ideal" availability level. Given that the data was generated from the original exponential mean time estimates of failure and repair times, the forecast at 60 days should not exceed the availability level of approximately 68%.

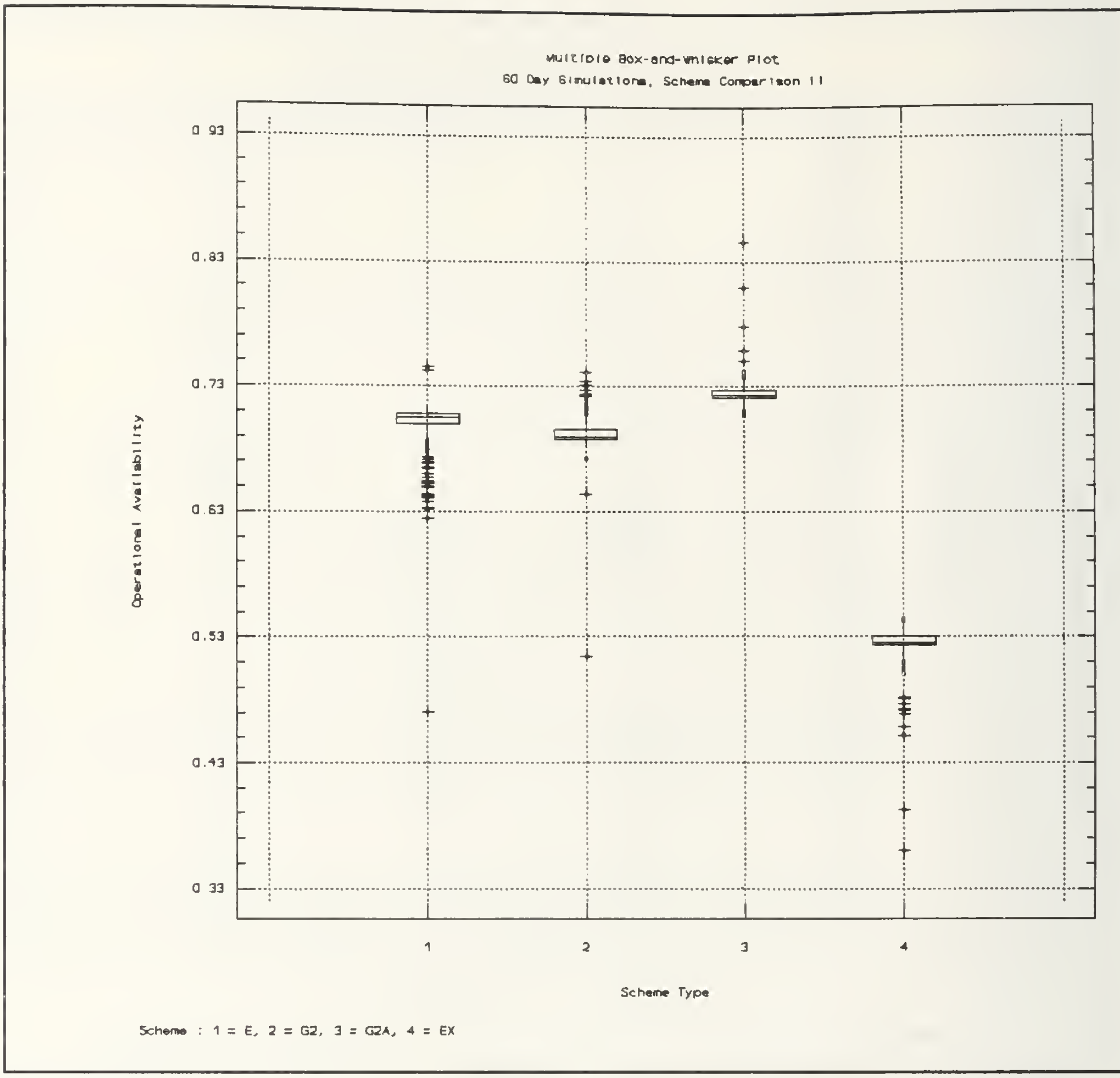

Figure 12

The updating methodology in taken in the context of schemes G2 and G2A indicate that the designed operation of the model would tend to underestimate the true availability given Gamma distributed failures or repairs by 16.8% in the case of scheme G2 and approximately 20% in scheme G2A's case.

These comparisons indicate that the operation of MIMMSIM with Bayesian updating performs adequately when the actual underlying failure or repair behavior of

equipment does not significantly deviate from exponential behavior. As the times of both failures and repairs begin to follow divergent paths, the accuracy of the predictors begins to degrade significantly.

B. MODEL VERIFICATION

Verification of the modeling approach would suggest applying the model to several actual data files from the MIMMS system, initiating the model accordingly, and comparing the results. At this time, the data necessary to implement this process is not available, due to a large influx of equipment into the maintenance cycle returning from the Gulf War. Given the prior designated 16-18 months worth of failure and repair dates, and roughly one quarter's worth of recent maintenance data, for two items, however, the MIMMSIM Prototype model can adequately survey verification procedures, with slight modifications.

For individual or two-item comparisons, this method would be ideal; the potential advantage of MIMMSIM, however, lies in its use to evaluate widely varying types of equipment *simultaneously*; the resultant analysis could lead to further studies regarding the interoperability of diverse systems and their impact on mission effectiveness.

C. BAYESIAN AND EMPIRICAL BAYES ANALYSIS

Bayesian analysis, particularly parametric empirical Bayesian (PEB) analysis, suggest themselves when failure or repair rates of different items are similar, but experience is limited; the opportunity to "fold in" available past information allows a better starting point, and greater confidence in the preliminary estimators.

In this study, failure and repair rates for each major end item λ_{ik} were assumed to follow a Gamma prior distribution. This prior distribution corresponds to assuming we have already performed a certain number of tests, and all were successful. It "corresponds" in the sense that you get exactly the same lower confidence limit values via the Bayesian methods and the Classical methods.[Ref. 15:p. 3] It provides a vehicle for folding in information on past performance, allowing less testing to be done subsequently to confirm the results.

The primary disadvantage of the classical Bayesian approach is that the prior distribution's parameters (i.e., α_{ik} and β_{ik}) are assumed to be known. This deficiency is surmounted in this study by the use of a parametric empirical Bayesian technique, in which the entire available data is used to estimate the parameters of the prior distribution. These estimates are then used in the standard Bayesian framework to compute the posterior and predictive distributions.

D. COMPUTATIONAL EXPERIENCE

The original goal of providing a model for use at a level capable of providing adequate information, was modified somewhat to take advantage of the power and flexibility of MODSIM II. While all elements necessary to operate MODSIM II are commercially available, the Marine Corps does not currently contract for models written in that language. Object-oriented programming is, however, expected to become the standard in the near future. The obvious utility alone of this medium should lend weight

to the process of implementing processes in more useful and powerful languages of this type.

The Prototype MIMMSIM model consists of 23 modules, which take approximately 15 minutes to compile on an IBM-compatible, 386-25 based personal computer, with 8 megabytes of RAM. Execution of the program takes from 2 minutes and 17 seconds for the 60 day suite simulation to run, to approximately 10 minutes for the 300 day simulation. Normal operation would entail daily predictions for possibly 100 items, and "suite" simulation of 400-600, at any given time. Delays would increase, but not markedly so.

The inherent structure of MODSIM II produces efficient process based simulation, while maintaining high standards of structured coding and flexibility. Unfortunately, the accessory programs necessary to run this version of MODSIM II (Microsoft C v.6.00A Compiler, OS/2 v.1.2, and MODSIM II v. 1.6, itself), require roughly 28 megabytes of hard disk space; additional memory is required for the MIMMSIM program.

Again, the upgrading and modification of this model should entail little analysis, if the goal is to expand the core of represented items. Only when the time comes to further develop the reliability structure of the model, should extensive analysis occur. Even then, the modular structure of MODSIM II would allow a new component to be added, without affecting overall model operation.

Finding the maximum likelihood estimators of α_{ik} and β_{ik} can be difficult, at times. The likelihood function is highly non-linear, and care must be taken to ensure adequate

convergence of the numerical routine used to approximate the MLE's. A preliminary study is done to reduce non-linearities prior to GAMS implementation.

VI. CONCLUSIONS AND RECOMMENDATIONS

In this thesis, an empirically based maintenance forecasting model was developed which interfaced with yet did not interfere with, the daily operations of an existing maintenance information system known as MIMMS. The model not only produces forecasts of predicted state time for items experiencing changes of state, based on each item's own individually developed reliability behavior estimators, but it also sets up and runs a large scale simulation with the goal of computing operational availability estimates for the ground combat commander. The estimates are statistically grounded in an empirically based methodology for updating and can produce tailored predictors for specific areas of concern.

The model can be best implemented in parallel with MIMMS by observing and collecting daily transaction traffic without impeding daily maintenance activities. In its optimal mode it can provide forecasts which can then be used to produce realistic maintenance resource allocation schemes. It can also provide commanders and maintenance personnel with data in the form of usage based maintenance factors to explain requirements. In higher echelons of maintenance, the program can serve as a quality control monitor to indicate changes in system performance should new or different techniques, materials, or personnel be employed in the maintenance effort. It could also be used to "seek out" the necessary areas for further study simply by a study of non-conformities in the output. An area of significant departure from norms could

indicate an area for concentration of maintenance or a section of the model requiring modification.

Further enhancements of the model should concentrate on initiating analysis efforts to ascertain the functional form of the failure and the repair processes. Failures should occur subject to either usage, location, or equipment specific factors. Fortunately, MIMMS captures this data, but it does so infrequently and oftentimes inaccurately. A concentration in the area of labor and costs of maintenance could also lead to further delineation of the processes in operation during the repair cycle. Further study should also be conducted into the dependence between failures and repairs, if any exists, to ensure proper functional relationships within the structure of the MIMMSIM model. Finally, follow-on studies should be conducted to validate the usefulness of the model in use with actual data during normal day to day operations, and also inline placement within the LFADS module of the MAGTF II/LOG AIS system.

The long range uses of the "by-products" of the model are varied. These "by-products", the parameter estimates themselves, should converge after time. The resulting estimates could represent information of use to planners when contracting to procure similar equipment, in support or criticism of a contractor, or as a rough estimate of system reliability. Parameter estimates could become a valuable information commodity. Army units could share similar databases to Marine units deploying to provisional areas, or vice versa.

A better measure of effectiveness should be developed to indicate availability of equipment. Operational availability does not portray the entire story of equipment usage,

particularly usage over specific time periods. The development of a sufficient indicator of equipment usage that the user can refer to for estimates relevant to his own situation would greatly advance the flexibility and power of the model.

Finally, studies should be performed to ascertain the effect of different maintenance methods on overall equipment system interoperability. This is a realistic factor in successful mission completion. Each mix of equipment systems produces its own unique capability to accomplish a mission. Maintenance plays a real factor in the decision making process, so the *quality* of the maintenance effort becomes vital. MIMMSIM could readily exploit this quality factor, if it could be measured.

The future of the armed services lies in the direction of doing more with less, and operations will involve fewer troops and more equipment. Consequently, the maintenance effort will increase. In order to fight more effectively, our armed forces will depend more and more on getting better information from existing systems and using this information more effectively. The method discussed in this study is one way of moving towards this goal without sacrificing quality for quantity.

APPENDIX A. GAMS SOURCE CODE FOR M.L.E. COMPUTATION

```
$TITLE MAXIMUM LIKELIHOOD ESTIMATION
$OFFUPPER
$ONTEXT
 PROGRAM TO DETERMINE MAXIMUM LIKELIHOOD ESTIMATES FOR
ALPHA    AND BETA BASED ON NUMBER OF FAILURES OR REPAIRS (N),
AND THE    TOTAL TIME SPENT IN THAT STATE (T)
$OFFTEXT

SETS
    J  Number of occurrences /1*10/;

SCALARS  ALPHAMAX   Max of alpha parameter /1/
         BETAMAX    Max of beta parameter /500/
         T          Total time of occurrence type/600/
         N          Number of occurrences /10/;

VARIABLES
         ALPHA      Max estimate of alpha shape param
         BETA       Max estimate of beta scale param
         LKHD       Likelihood value;

POSITIVE VARIABLE ALPHA, BETA;

EQUATIONS
         ML         MAXIMUM LIKELIHOOD EQN
         ACON       ALPHA CONSTRAINT
         BCON       BETA CONSTRAINT;

ML .. LKHD =E= ((PROD
(J,(ORD(J))))*(PROD(J,(N-(ORD(J))+ALPHA)))*
((BETA/(T+BETA))**ALPHA)*((T/(T+BETA))**N));

ACON .. ALPHA =L= ALPHAMAX;

BCON .. BETA =L= BETAMAX;

MODEL LIKELIHOOD /ALL/;
```

```
SOLVE LIKELIHOOD USING NLP MAXIMIZING LKHD;

DISPLAY ALPHA.L, BETA.L;
```

APPENDIX B. SAMPLE INPUT FILES

File "testend.dat" : Contains initialization information on equipment items. Read in as 1 column.

A1000	B2000
A1	B1
UP	DOWN
NO	NO
91070	91063
NO	NO
31	4
A1000	B2000
A2	B2
DOWN	DOWN
YES	YES
91072	91079
NO	NO
15	42
A1000	B2000
A3	B3
DOWN	UP
NO	NO
91071	91066
NO	NO
23	11

File "trans1.dat" : Contains first "day's transactions.

```
91080
9
C
A1000
A2
91072
M

91080
0
C
B2000
B1
91063
M

91080
0
A
A1000
A1
91070
N
```

File "trans2.dat" : Contains second "day's" transactions.

```
91087
9
C
A1000
A3
91071
N

91087
0
A
B2000
B3
91066
N

91087
0
A
A1000
A2
91087
M

91087
0
C
A1000
A1
91087
M
```

File "trans3.dat" : Contains third "day's" transactions.

```
91091
0
C
B2000
B2
91091
N

91091
9
C
A1000
A2
91091
M
```

File "seeds.dat" : Contains seeds used in model.

```
80
  2000627
 72637153
148928099
137599518
 49312791
307220232
173186061
167073533
190563346
 22646660
207024957
104827570
 72854541
 77348606
131504639
 91969108
129826033
110307610
120229097
102542701
 85250373
 16619264
128786727
199704973
 39364491
 64009090
193937202
149215665
138551682
121449627
 89888577
129637518
 34270853
 46539910
150117532
165567359
110320053
103177493
548930245
332107876
```

212109845
37461113
484673212
190357605
122436784
193357605
72637053
148428099
137590518
49315791
207220232
173186021
167083533
190163346
72646660
207924957
104827580
74854541
77368606
138504639
61969108
129829033
110007610
120229997
102842701
85550373
17619264
118786727
199704913
39304491
64069090
193934202
159215665
138551692
181449627
89858577
129537518
24270853
46539010
150187532
165067359
110320553
100177493

546930245
332100876
221109845
34761113
444673212
2010627
122476784

APPENDIX C. SAMPLE OUTPUT FILES

A. DAILY OUTPUT

File : "day1pred.txt"

Daily Predictions

Today's Date:91080

		Daily Prediction	
Item ID #	Serial Number	UP	DOWN
A1000	A2	2.33	
B2000	B1		1.07

File : "day2pred.txt"

Daily Predictions

Today's Date:91087

		Daily Prediction	
Item ID #	Serial Number	UP	DOWN
A1000	A1		6.07
A1000	A2		7.34

File : "day3pred.txt"

Daily Predictions

Today's Date:91091

		Daily Prediction	
Item ID #	Serial Number	UP	DOWN
A1000	A2	2.85	
B2000	B2	15.08	

B. SIMULATION OUTPUT

File : "simout1.txt"

Suite Simulation

FROM	TO	FOR (days)
91080	92015	300

Results :

Item ID #	Operational Availability (%)
A1000	67.656
B2000	88.575

APPENDIX D. MIMMSIM MODSIM II SOURCE CODE

```
MAIN MODULE mimmsim;

{***********************************************************

   MAIN MODULE MIMMSIM
   AUTHOR :  B. F. MIMMS        DATE WRITTEN :   08 July 92
             CAPT  USMC          LAST MODIFIED :  25 Aug  92

   DESCRIPTION :  Main module of program. Initiates all
            activities, coordinates information flow,
            and program control; sets up and routes
            "daily" transaction traffic; initializes
            and starts simulation runs.

***********************************************************}


   FROM global          IMPORT listing, roster, member,
                         start, stop, firstTime,
                         update,done1,done2,
                         done3,done4,q,
                         AparameterQ,BparameterQ,
                         SArrayType, fileArray,
                         i,n,todaysDate,
                         alphaUpLambda,
                         alphaDownLambda,
                         betaUpLambda,
                         betaDownLambda,
                         quitTime,quitDate,
                         simListing;
   FROM queueL          IMPORT queueListObj;
   FROM proced          IMPORT initializeItems, readXact,
                         upDate, julianDiff,
                         setupSim,dailyOutput;
   FROM output          IMPORT simOutput;
   FROM param           IMPORT initializeParams;
   FROM predict         IMPORT readSeeds;
   FROM Debug           IMPORT TraceStream;
```

```
FROM debugRn         IMPORT SetUpD;
FROM SimMod          IMPORT StartSimulation,
                     StopSimulation,
                     ResetSimTime;
FROM simulat        IMPORT AuptimeStats,AdowntimeStats,
                     BuptimeStats,BdowntimeStats,
                     commenceSim,resetStats;


VAR
    file1,file2,file3 : STRING;
    day               : INTEGER;
    quitDate1,quitDate2,quitDate3 : REAL;
BEGIN

    SetUpD(FALSE);
    firstTime := "YES";
    update    := "NO";
    done1     := "NO";
    done2     := "NO";
    done3     := "NO";
    done4     := "NO";
    file1    := "trans1.dat";
    file2    := "trans2.dat";
    file3    := "trans3.dat";
    n        := 1;

    initializeItems (listing);

    initializeParams (AparameterQ, BparameterQ);

    readSeeds();
{READ TRANSACTIONS FROM 1ST FILE}

    day := 1;
    readXact (file1,roster);

    upDate (listing, member,roster,AparameterQ,
           BparameterQ,n,update,done1,
           done2,done3,done4,firstTime,todaysDate);

    dailyOutput(listing,todaysDate,day);
```

```
        setupSim(n,AparameterQ,BparameterQ,
                todaysDate,listing,alphaUpLambda,
                alphaDownLambda,betaUpLambda,
                betaDownLambda,quitTime,quitDate1,
                simListing,day);
        FOR i := 1 TO 500
        commenceSim (alphaUpLambda,alphaDownLambda,
                  betaUpLambda,betaDownLambda,
                  quitTime,simListing);
        StartSimulation;
        ResetSimTime(0.0);
        END FOR;
        simOutput(todaysDate,quitDate1,quitTime,day);

        StopSimulation;
        resetStats;

{READ TRANSACTIONS FROM 2ND FILE}
        day := 2;
        readXact (file2,roster);

        upDate (listing, member,roster,AparameterQ,
               BparameterQ,n,update,done1,
               done2,done3,done4,firstTime,todaysDate);

        dailyOutput(listing,todaysDate,day);

        setupSim(n,AparameterQ,BparameterQ,
                todaysDate,listing,alphaUpLambda,
                alphaDownLambda,betaUpLambda,
                betaDownLambda,quitTime,quitDate1,
                simListing,day);
        FOR i := 1 TO 500
        commenceSim (alphaUpLambda,alphaDownLambda,
                  betaUpLambda,betaDownLambda,
                  quitTime,simListing);
        StartSimulation;
        ResetSimTime(0.0);
        END FOR;
        simOutput(todaysDate,quitDate1,quitTime,day);

        StopSimulation;
        resetStats;
```

```
{READ TRANSACTIONS FROM 3RD FILE}
        day := 3;
        readXact (file3,roster);

        upDate (listing, member,roster,AparameterQ,
               BparameterQ,n,update,done1,
               done2,done3,done4,firstTime,todaysDate);

        dailyOutput(listing,todaysDate,day);

        setupSim(n,AparameterQ,BparameterQ,
                todaysDate,listing,alphaUpLambda,
                alphaDownLambda,betaUpLambda,
                betaDownLambda,quitTime,quitDate1,
                simListing,day);
        FOR i := 1 TO 500
        commenceSim (alphaUpLambda,alphaDownLambda,
                    betaUpLambda,betaDownLambda,
                    quitTime,simListing);
        StartSimulation;
        ResetSimTime(0.0);
        END FOR;
        simOutput(todaysDate,quitDate1,quitTime,day);

        StopSimulation;
        resetStats;

END {MAIN} MODULE {mimmsim}.
```

```
DEFINITION MODULE endItem;

{******************************************************************

    MODULE NAME : DendItem          DATE WRITTEN :  08 July 92
    AUTHOR :      B. F. Mimms       LAST MODIFIED : 29 July 92
                  CAPT   USMC

    DESCRIPTION : Defines and implements end item type and
          methods.

******************************************************************}

    FROM IOMod       IMPORT ALL FileUseType, StreamObj;
  FROM Debug       IMPORT TraceStream;

  TYPE

    endItemObj = OBJECT
       idno        : STRING;
       sernum       : STRING;
       updown       : STRING;
       deadline     : STRING;
       statdate    : STRING;
       predict     : STRING;
       predval      : STRING;
       simpredval  : REAL;
       ASK METHOD readData (IN inputStrm : StreamObj);
         ASK METHOD changeid (IN newid  :  STRING);
         ASK METHOD changesn (IN newser : STRING);
       ASK METHOD changeud (IN newud : STRING);
       ASK METHOD changedl (IN newdl : STRING);
       ASK METHOD changestat (IN newdate : STRING);
       ASK METHOD changepred (IN newpred : STRING);
       ASK METHOD changesimpred (IN newsimpred :REAL);
    END OBJECT;

END {DEFINITION} MODULE {endItem}.
```

```
IMPLEMENTATION MODULE endItem;

{*************************************************************

    MODULE NAME : IendItem          DATE WRITTEN :  08 July 92
    AUTHOR :      B. F. Mimms       LAST MODIFIED : 29 July 92
                  CAPT   USMC

    DESCRIPTION : Defines and implements end item type and
                  methods.

*************************************************************}

    FROM IOMod      IMPORT ALL FileUseType, StreamObj;
    FROM Debug      IMPORT TraceStream;

  OBJECT endItemObj;

      ASK METHOD readData (IN instrm  : StreamObj);
      BEGIN

          idno := "";

          WHILE (idno = "")
              ASK instrm TO ReadLine (idno);
          END WHILE;

          ASK instrm TO ReadLine (sernum);
          ASK instrm TO ReadLine (updown);
          ASK instrm TO ReadLine (deadline);
          ASK instrm TO ReadLine (statdate);
          ASK instrm TO ReadLine (predict);
          ASK instrm TO ReadLine (predval);

      END {readData} METHOD;

      ASK METHOD changeid (IN newid : STRING);
      BEGIN
          idno := newid;
      END METHOD {changeid};
```

```
    ASK METHOD changesn (IN newser : STRING);
    BEGIN
        sernum := newser;
    END METHOD {changesn};

    ASK METHOD changeud (IN newud : STRING);
    BEGIN
        updown := newud;
    END METHOD {changeud};

    ASK METHOD changedl (IN newdl : STRING);
    BEGIN
        deadline := newdl;
    END METHOD {changedl};

    ASK METHOD changestat (IN newdate : STRING);
    BEGIN
      statdate := newdate;
    END METHOD {changestat};

    ASK METHOD changepred (IN newpred : STRING);
    BEGIN
      predict := newpred;
    END METHOD {changepred};

    ASK METHOD changesimpred(IN newsimpred : REAL);
    BEGIN
     simpredval := newsimpred;
    END METHOD;


  END {endItem} OBJECT;
END {IMPLEMENTATION} MODULE.
```

```
DEFINITION MODULE transac;

{************************************************************

     MODULE NAME : Dtransac          DATE WRITTEN :    8 July 92
     AUTHOR :      B. F. Mimms       LAST MODIFIED : 10 July 92
                   CAPT   USMC

     DESCRIPTION : Contains information on transaction objects

************************************************************

     FROM IOMod      IMPORT StreamObj;
     FROM Debug      IMPORT TraceStream;

   TYPE
         transactionObj  = OBJECT
                date         :  STRING;
                transcode    :  STRING;
                transtype    :  STRING;
                idno         :  STRING;
                sernum       :  STRING;
                dcd          :  STRING;
                cat          :  STRING;
                ASK METHOD readData (IN inputStrm :
                           StreamObj);
   END OBJECT;

END {DEFINITION} MODULE {transac}.
```

```
IMPLEMENTATION MODULE transac;

{**********************************************************

     MODULE NAME : Itransac         DATE WRITTEN :   8 July 92
     AUTHOR :       B. F. Mimms      LAST MODIFIED : 10 July 92
                    CAPT    USMC

     DESCRIPTION : Contains information on transaction objects

**********************************************************

     FROM IOMod     IMPORT StreamObj;
     FROM Debug     IMPORT TraceStream;

      OBJECT transactionObj;

           ASK METHOD readData (IN instrm  : StreamObj);
           BEGIN
                date := "";
                WHILE  (date = "")
                       ASK instrm TO ReadLine (date);
                END WHILE;

                ASK instrm TO ReadLine (transcode);
                ASK instrm TO ReadLine (transtype);
                ASK instrm TO ReadLine (idno);
                ASK instrm TO ReadLine (sernum);
                ASK instrm TO ReadLine (dcd);
                ASK instrm TO ReadLine (cat);
           END METHOD {readData};

      END OBJECT {transactionObj};

END {IMPLEMENTATION} MODULE {transac}.
```

```
DEFINITION MODULE global;

{*************************************************************

    MODULE NAME : Dglobal          DATE WRITTEN   :  08 July 92
    AUTHOR :      B. F. Mimms      LAST MODIFIED :  29 July 92
                  CAPT   USMC

    DESCRIPTION :  Contains global variables used throughout
                the model.

*************************************************************

   FROM endItem     IMPORT endItemObj;
   FROM transac     IMPORT transactionObj;
   FROM queueL      IMPORT queueListObj;
   FROM param       IMPORT A1000Obj, B2000Obj;
   FROM Debug       IMPORT TraceStream;

   TYPE
      seedArrayType  = ARRAY INTEGER OF INTEGER;
      SArrayType     = ARRAY INTEGER OF STRING;

   VAR
      member            : endItemObj;
      transact          : transactionObj;
      listing, roster,
      AparameterQ,
      BparameterQ       : queueListObj;
      parameterA        : A1000Obj;
      parameterB        : B2000Obj;
      date, transcode,
      transtype,idnum,
      serno,dcd,cat     : STRING;
      idno,sernum,
      updown,deadline,
      statdate,predict  : STRING;
      start, stop,
      jDiff, todaysDate,
      newlambda,newUptime,
      newDowntime       : REAL;
      done1,done2,done3,
      done4,firstTime,
```

```
        update, newpred    : STRING;
        qtr, prevQtr,
        qDiff, q, qtrCheck,
        utauCheck,unumCheck,
        n,i                : INTEGER;
        dtauCheck,dnumCheck,
        utauHold,unumHold,
        dtauHold,dnumHold  : INTEGER;
        seedArray          : seedArrayType;
        fileArray          : SArrayType;
        alphaUpLambda,
        alphaDownLambda,
        betaUpLambda,
        betaDownLambda,
        quitTime,
        quitDate,pred      : REAL;
        simListing         : queueListObj;

END {DEFINITION} MODULE {global}.


IMPLEMENTATION MODULE global;

{*************************************************************

    MODULE NAME : Dglobal        DATE WRITTEN  :  08 July 92
    AUTHOR :      B. F. Mimms    LAST MODIFIED : 29 July 92
                  CAPT   USMC

    DESCRIPTION :  Contains global variables used throughout
              the model.

************************************************************

END {IMPLEMENTATION} MODULE {global}.
```

```
DEFINITION MODULE queueL;

{***********************************************************

    MODULE NAME : Dqueuel        DATE WRITTEN :    8 July 92
    AUTHOR :      B. F. Mimms    LAST MODIFIED :  12 July 92
                  CAPT   USMC

    DESCRIPTION : Contains information on queueObj's in use in
                  program

***********************************************************

   FROM GrpMod IMPORT QueueObj;
   FROM Debug  IMPORT TraceStream;

   TYPE
      queueListObj = OBJECT (QueueObj)
      END OBJECT;

END {DEFINITION} MODULE {queueL}.


IMPLEMENTATION MODULE queueL;

{***********************************************************

    MODULE NAME : Iqueuel        DATE WRITTEN :    8 July 92
    AUTHOR :      B. F. Mimms    LAST MODIFIED :  12 July 92
                  CAPT   USMC

    DESCRIPTION : Contains information on queueObj's in use in
                  program
***********************************************************

   FROM GrpMod IMPORT QueueObj;
   FROM Debug  IMPORT TraceStream;

   OBJECT queueListObj;

   END OBJECT;
END {IMPLEMENTATION}  MODULE {queueL}.
```

```
DEFINITION MODULE debugRn;

{***********************************************************

     MODULE NAME :  DdebugRn      DATE WRITTEN :   28 July 92
     AUTHOR      : J. Judy        LAST MODIFIED  : 28 July 92
                   CPT USA
     MODIFIED BY :  B. F. Mimms
                    CAPT   USMC

     DESCRIPTION : Contains TraceStream used for debugging
            of program

***********************************************************}

  PROCEDURE SetUpD (IN TraceOn : BOOLEAN);

END {DEFINITION} MODULE {deBug}.


IMPLEMENTATION MODULE debugRn;

{***********************************************************

     MODULE NAME :  IdebugRn      DATE WRITTEN :   28 July 92
     AUTHOR      : J. Judy        LAST MODIFIED  : 28 July 92
                   CPT USA
     MODIFIED BY :  B. F. Mimms
                    CAPT   USMC

  DESCRIPTION : Contains TraceStream used for debugging
            of program

***********************************************************}

   FROM IOMod    IMPORT FileUseType (Output);
   FROM Debug    IMPORT TraceStream;
   FROM UtilMod  IMPORT DateTime;

   PROCEDURE SetUpD (IN TraceOn : BOOLEAN);
```

```
VAR
  DT : STRING;

BEGIN
   NEW(TraceStream);
   ASK TraceStream TO Open ("debug.out", Output);
         DateTime(DT);
   ASK TraceStream TO WriteString(DT);
   ASK TraceStream TO WriteLn;
   ASK TraceStream TO WriteLn;
   ASK TraceStream TO WriteLn;

   IF TraceOn
       ASK TraceStream TO TraceOn;
       OUTPUT("--------TRACE ON----------");
       ASK TraceStream TO WriteString("Initially, trace
      is on.");
       ASK TraceStream TO WriteLn;
    ELSE
       ASK TraceStream TO TraceOff;
       ASK TraceStream TO WriteString("Initially, trace
      is off.");
       ASK TraceStream TO WriteLn;
     END IF;
 END PROCEDURE;
END {IMPLEMENTATION} MODULE {deBug}.
```

```
DEFINITION MODULE proced;

{****************************************************************

    MODULE NAME : Dproced        DATE WRITTEN :  08 July 92
    AUTHOR :      B. F. Mimms    LAST MODIFIED : 08 Aug 92
                  CAPT   USMC

    DESCRIPTION : Contains most procedural processes invoked
            throughout the program.

****************************************************************}

   FROM endItem       IMPORT endItemObj;
   FROM transac       IMPORT transactionObj;
   FROM queueL        IMPORT queueListObj;
   FROM IOMod         IMPORT ALL FileUseType, StreamObj;
   FROM param         IMPORT TimeObj;
   FROM global        IMPORT n,pred, update,done1,done2,
                       done3,done4,firstTime;
   FROM simulat       IMPORT initializeSuiteSimulation;
   FROM Debug         IMPORT TraceStream;

   PROCEDURE initializeItems (OUT listing : queueListObj);
   PROCEDURE readXact (IN file: STRING;OUT roster  :
                         queueListObj);
   PROCEDURE upDate (INOUT listing :queueListObj;
              INOUT member : endItemObj;
              IN roster : queueListObj;
              OUT AparameterQ, BparameterQ :
                    queueListObj;INOUT n : INTEGER;
              INOUT update,done1,done2,
              done3,done4,firstTime: STRING;
              OUT todaysDate : REAL);
   PROCEDURE dailyOutput (IN listing : queueListObj; IN
                         todaysDate : REAL;IN day :INTEGER);
   PROCEDURE julianDiff (IN start, stop : REAL; OUT jDiff :REAL);
   PROCEDURE setupSim (INOUT n : INTEGER;IN AparameterQ,
               BparameterQ :queueListObj;
               IN todaysDate : REAL;
               IN listing : queueListObj;
               OUT alphaUpLambda,
               alphaDownLambda,
```

```
                    betaUpLambda,
                    betaDownLambda,quitTime,
                    quitDate : REAL; OUT simListing :
                    queueListObj; IN day : INTEGER);
END {DEFINITION} MODULE {proced}.

IMPLEMENTATION MODULE proced;

{*************************************************************

    MODULE NAME : Iproced       DATE WRITTEN :  08 July 92
    AUTHOR :      B. F. Mimms   LAST MODIFIED : 08 Aug 92
                  CAPT   USMC

    DESCRIPTION : Contains most procedural processes invoked
                  throughout the program.

*************************************************************}

   FROM endItem      IMPORT endItemObj;
   FROM transac      IMPORT transactionObj;
   FROM queueL       IMPORT queueListObj;
   FROM IOMod        IMPORT ALL FileUseType, StreamObj;
   FROM global       IMPORT member, transact, listing,
                     roster, date, transcode,
                     transtype,idnum,serno,
                     dcd, cat,idno,sernum,
                     updown, deadline, statdate,
                     predict, start, stop,jDiff,
                     todaysDate,qtr,newUptime,
                     newDowntime,parameterA,
                     parameterB,n, pred,update,
                     done1,done2,done3,done4,
                     firstTime;
   FROM upDater      IMPORT checkUpdate;
   FROM param        IMPORT TimeObj, dailyAUpdate,
                     dailyBUpdate, A1000Obj,
                     B2000Obj;
   FROM predict      IMPORT predictUp, predictDown;
   FROM simulat      IMPORT initializeSuiteSimulation;
   FROM Debug        IMPORT TraceStream;
```

```
{*********************************************************}
   PROCEDURE initializeItems (OUT listing : queueListObj);
{*********************************************************}

   VAR
      strmIn    : StreamObj;

   BEGIN

      NEW (listing);
      NEW (strmIn);

      ASK strmIn TO Open ("testend.dat", Input);
      WHILE NOT strmIn.eof

         NEW (member);
         ASK member TO readData(strmIn);
         ASK listing TO Add (member);

      END WHILE;

      ASK strmIn TO Close;

   END {initializeItems} PROCEDURE;

{*********************************************************}
   PROCEDURE readXact (IN file : STRING; OUT roster :
 queueListObj);
{*********************************************************}

    VAR
      strmIn   : StreamObj;

    BEGIN

      NEW (roster);
      NEW (strmIn);

      ASK strmIn TO Open (file, Input);
      WHILE NOT strmIn.eof

         NEW (transact);
         ASK transact TO readData(strmIn);
```

```
        ASK roster TO Add (transact);
    END WHILE;

    ASK strmIn TO Close;

END {readXacts} PROCEDURE;

{**********************************************************}
    PROCEDURE upDate (INOUT listing : queueListObj ;
                  INOUT member : endItemObj;
                  IN roster : queueListObj;
                  OUT AparameterQ, BparameterQ:
                  queueListObj; INOUT n : INTEGER;
                  INOUT update,done1,done2,done3,
                  done4,firstTime: STRING;
                  OUT todaysDate : REAL);
{**********************************************************}

    VAR
        TimeCount : TimeObj;

    BEGIN
        NEW(TimeCount);
        transact    := ASK roster First();
        todaysDate  := STRTOREAL(transact.date);
        checkUpdate(todaysDate, qtr, AparameterQ,
                BparameterQ,update,done1,done2,
                done3,done4,firstTime);

        member := ASK listing First();
        WHILE member <> NILOBJ
            transact := ASK roster First();
            WHILE transact <> NILOBJ
                ASK member TO changesimpred(0.0);

                date       := transact.date;
                transcode  := transact.transcode;
                transtyp   := transact.transtype;
                idnum      := transact.idno;
                serno      := transact.sernum;
                dcd        := transact.dcd;
                cat        := transact.cat;
                idno       := member.idno;
```

```
                    sernum      := member.sernum;
                    updown      := member.updown;
                    deadline    := member.deadline;
                    statdate    := member.statdate;
                    predict     := member.predict;

        IF idnum = idno
          IF serno = sernum
            IF transcode = "0"
{------------------------------------------------------------
          O/A TRANSACTION
------------------------------------------------------------}
                    IF transtype = "A"
                        IF updown = "UP"
                            ASK member TO changeud ("DOWN");
                            ASK member TO changedl ("NO");
                              IF cat = "M"
                        ASK member TO changedl("YES");
                        ASK member TO changepred("YES");
                              julianDiff (STRTOREAL
                              (statdate),STRTOREAL(dcd),newUptime);
                              IF idno = "A1000"
                                ASK TimeCount TO sumAUpTime(newUptime);
                              ELSIF idno = "B2000"
                                ASK TimeCount TO sumBUpTime(newUptime);
                              END IF;
                                ASK member TO changestat(dcd);
                      END IF;
                    END IF;
                  ELSE
{------------------------------------------------------------
          O/C TRANSACTION
------------------------------------------------------------}
                      IF updown = "DOWN"
                      IF deadline = "YES"
                        IF cat <> "M"{Out of Repair}
                            julianDiff (STRTOREAL
                                  (statdate),
                                  STRTOREAL(date),
                                  newDowntime);
                            IF idno = "A1000"
                              ASKTimeCountTOsumADownTime(newDowntime);
                            ELSIF idno = "B2000"
```

```
                    ASKTimeCountTOsumBDownTime(newDowntime);
                  END IF;
                  ASK member TO changestat(date);
                  ASK member TO changedl ("NO");
                    ASK member TO changeud ("UP");
                   ASK member TO changepred("YES");
              ELSE
                  ASK member TO changestat (dcd);
              END IF;
            ELSE
              IF cat = "M"{Into Repair}
                ASK member TO changedl ("YES");
                  ASK member TO changeud ("DOWN");
                 ASK member TO changepred ("YES");
                  julianDiff (STRTOREAL
                         (statdate),
                         STRTOREAL(dcd),
                         newUptime);
                  IF idno = "A1000"
                    ASK TimeCount TO sumAUpTime(newUptime);
                  ELSIF idno = "B2000"
                    ASK TimeCount TO sumBUpTime(newUptime);
                  END IF;
                  ASK member TO changestat ("dcd");
              END IF;
          END IF;
        END IF;
      END IF;
{---------------------------------------------------------
     9/C TRANSACTION
---------------------------------------------------------}
       ELSIF transcode = "9"
         IF deadline = "YES"
           julianDiff (STRTOREAL(statdate),
                   STRTOREAL(date),
                   newDowntime);
           IF idno = "A1000"
             ASK TimeCount TO sumADownTime
                      (newDowntime);
           ELSIF idno = "B2000"
             ASK TimeCount TO sumBDownTime
                      (newDowntime);
           END IF;
```

```
                ASK member TO changestat (date);
                ASK member TO changeud ("UP");
                ASK member TO changepred ("YES");
                ASK member TO changedl ("NO");
            ELSE
                        ASK member TO changeud ("DOWN");
            END IF;
           END IF;
      END IF;
    END IF;

       transact := ASK roster Next(transact);
       END WHILE;
       member := ASK listing Next(member);
       END WHILE;

       parameterA := ASK AparameterQ First();
       WHILE parameterA <> NILOBJ

            IF qtr = parameterA.qtr

               dailyAUpdate (qtr,TimeCount,parameterA);

            END IF;
         parameterA := ASK AparameterQ Next(parameterA);
         END WHILE;


         parameterB := ASK BparameterQ First();
         WHILE parameterB <> NILOBJ

            IF qtr = parameterB.qtr

               dailyBUpdate (qtr,TimeCount,parameterB);

            END IF;
          parameterB := ASK BparameterQ Next(parameterB);
          END WHILE;

          member := ASK listing First();
          WHILE member <> NILOBJ
              IF member.predict = "YES"
                 IF member.updown = "UP"
```

```
            idno := member.idno;
            predictUp (n, qtr,AparameterQ,
                       BparameterQ, idno,
                       pred);
            ASK member TO changesimpred(pred);
          ELSE
            idno := member.idno;
            predictDown (n, qtr, AparameterQ,
                         BparameterQ, idno,
                         pred);
            ASK member TO changesimpred(pred);
          END IF;
       END IF;
      member := ASK listing Next(member);
      INC (n);
   END WHILE;


END PROCEDURE {upDate};

{*****************************************************}
PROCEDURE dailyOutput(IN listing : queueListObj;
                      IN todaysDate : REAL;
                      IN day : INTEGER);
{*****************************************************}

VAR
    Streamer    : StreamObj;
    idno,sernum : STRING;
    pred        : REAL;
CONST
    headDaily  = "   Daily Predictions ";
    headDaily1 = "    Daily Prediction";
    headDaily2 = "Item ID #  Serial Number  UP DOWN";
    headDaily3 = "Today's Date:";
    dailyFormatDate = "*****";
    dailyFormatUp = "*****   **        **.** ";
    dailyFormatDwn = "*****   **        **.** ";
BEGIN
    NEW(Streamer);

    IF day = 1
       ASK Streamer TO Open("day1pred.txt",Output);
```

```
    ELSIF day = 2
        ASK Streamer TO Open("day2pred.txt",Output);
    ELSIF day =3
        ASK Streamer TO Open("day3pred.txt",Output);
    END IF;

    ASK Streamer TO WriteString(headDaily);
    ASK Streamer TO WriteLn;
    ASK Streamer TO WriteString(headDaily3);
    ASK Streamer TO WriteString(SPRINT(todaysDate) WITH dailyFormatDate);
    ASK Streamer TO WriteLn;
    ASK Streamer TO WriteString(headDaily1);
    ASK Streamer TO WriteLn;
    ASK Streamer TO WriteString(headDaily2);
    ASK Streamer TO WriteLn;
    member := ASK listing First();
    WHILE member <> NILOBJ
        idno := member.idno;
        sernum := member.sernum;
        pred := member.simpredval;
        IF member.predict = "YES"
        IF member.updown = "UP"
            ASK Streamer TO WriteString(SPRINT(idno,sernum,pred)
                        WITH dailyFormatUp);
            ASK Streamer TO WriteLn;
            ASK member TO changepred("NO");
        ELSE
            ASK Streamer TO WriteString(SPRINT(idno,sernum,pred)
                WITH dailyFormatDwn);
            ASK Streamer TO WriteLn;
            ASK member TO changepred("NO");
        END IF;
      END IF;
        member := ASK listing Next(member);
    END WHILE;
    ASK Streamer TO Close;
    DISPOSE(Streamer);
END PROCEDURE;
```

```
{*****************************************************************}
   PROCEDURE julianDiff (IN start, stop : REAL; OUT jDiff:
                    REAL );
{*****************************************************************}

    VAR
        actualDiff,  x : REAL;

    CONST
        shift = 635.0;

    BEGIN
        actualDiff := stop - start;
        x := FLOAT( ROUND (actualDiff / 1000.0));
        jDiff := actualDiff - (x * shift);

    END PROCEDURE {julianDiff};

{***************************************************************}
    PROCEDURE setupSim( INOUT n : INTEGER;IN AparameterQ,
                  BparameterQ :  queueListObj;
                  IN todaysDate : REAL;
                  IN listing : queueListObj;
                  OUT alphaUpLambda,alphaDownLambda,
                  betaUpLambda, betaDownLambda,
                  quitTime, quitDate : REAL;
                  OUT simListing : queueListObj;
                  IN day : INTEGER);
{***************************************************************}

    VAR
        qtrArec          : A1000Obj;
        qtrBrec          : B2000Obj;
        qtrCheck         : INTEGER;
        quitDate1,
        quitDate2,
        quitDate3        : REAL;

    BEGIN

        IF day = 1
           quitDate1 := 91140.0; { 60 Day simulation}
          {quitDate1 := 91230.0;}{150 Day simulation}
```

```
          {quitDate1 := 92015.0;}{300 Day simulation}
           quitDate := quitDate1;
           julianDiff(todaysDate,quitDate1,quitTime);
       ELSIF day =2
           quitDate2 := 91147.0; { 60 Day simulation}
          {quitDate2 := 91237.0;}{150 Day simulation}
          {quitDate2 := 92022.0;}{300 Day simulation}
           quitDate := quitDate2;
           julianDiff(todaysDate,quitDate2,quitTime);
       ELSIF day =3
           quitDate3 := 91151.0; { 60 Day simulation}
          {quitDate3 := 91241.0;}{150 Day simulation}
          {quitDate3 := 92026.0;}{300 Day simulation}
           quitDate := quitDate3;
           julianDiff(todaysDate,quitDate3,quitTime);
       END IF;
           initializeSuiteSimulation (n,listing,simListing);

           qtrArec  := ASK AparameterQ First();
           WHILE qtrArec <> NILOBJ
                qtrCheck  := qtrArec.qtr;
                IF qtr = qtrCheck
                    alphaUpLambda := qtrArec.ulambda;
                    alphaDownLambda := qtrArec.dlambda;
                END IF;

           qtrArec := ASK AparameterQ Next(qtrArec);
           END WHILE;

           qtrBrec := ASK BparameterQ First();
           WHILE qtrBrec <> NILOBJ
                qtrCheck   := qtrBrec.qtr;
                IF qtr = qtrCheck
                   betaUpLambda := qtrBrec.ulambda;
                   betaDownLambda := qtrBrec.dlambda;
                END IF;

           qtrBrec := ASK BparameterQ Next(qtrBrec);
           END WHILE;
           OUTPUT("Ready To Commence Simulation....");

   END PROCEDURE;
END {IMPLEMENTATION} MODULE {proced}.
```

```
DEFINITION MODULE param;

{*************************************************************

    MODULE NAME : Dparam          DATE WRITTEN :  13 July 92
    AUTHOR :  B. F. Mimms         LAST MODIFIED :  29 July 92
              CAPT   USMC

    DESCRIPTION : Contains parameter queues, and updating timer object. Contains
                 methods and procedures to reference and update
                 parameters on a daily and quarterly basis.

*************************************************************}

   FROM queueL        IMPORT queueListObj;
   FROM Debug         IMPORT TraceStream;

   TYPE

     A1000Obj  = OBJECT
           qtr       : INTEGER;
           ulambda  : REAL;
           utau      : INTEGER;
           unum      : INTEGER;
           dlambda  : REAL;
           dtau      : INTEGER;
           dnum      : INTEGER;
           ASK METHOD changeQtr (IN newqtr :
              INTEGER);
           ASK METHOD changeUlambda (IN newulambda:
              REAL);
           ASK METHOD changeUtau (IN newutau :
              INTEGER);
           ASK METHOD changeUnum (IN newunum :
              INTEGER);
           ASK METHOD changeDlambda (IN newdlambda :
              REAL);
           ASK METHOD changeDtau (IN newdtau :
              INTEGER);
           ASK METHOD changeDnum (IN newdnum :
              INTEGER);
     END OBJECT;
```

```
B2000Obj = OBJECT;
        qtr     : INTEGER;
        ulambda : REAL;
        utau    : INTEGER;
        unum    : INTEGER;
        dlambda : REAL;
        dtau    : INTEGER;
        dnum    : INTEGER;
        ASK METHOD changeQtr (IN newqtr :
           INTEGER);
        ASK METHOD changeUlambda (IN newulambda:
           REAL);
        ASK METHOD changeUtau (IN newutau :
           INTEGER);
        ASK METHOD changeUnum (IN newunum :
           INTEGER);
        ASK METHOD changeDlambda(IN newdlambda :
           REAL);
        ASK METHOD changeDtau (IN newdtau :
           INTEGER);
        ASK METHOD changeDnum (IN newdnum :
           INTEGER);
END OBJECT;

TimeObj = OBJECT;
        A1000UpTime      : INTEGER;
        A1000CountUp     : INTEGER;
        A1000DownTime    : INTEGER;
        A1000CountDown   : INTEGER;
        B2000UpTime      : INTEGER;
        B2000CountUp     : INTEGER;
        B2000DownTime    : INTEGER;
        B2000CountDown   : INTEGER;
        ASK METHOD sumAUpTime (IN newUptime :
           REAL);
        ASK METHOD sumADownTime (IN newDowntime :
           REAL);
        ASK METHOD sumBUpTime (IN newUptime :
           REAL);
        ASK METHOD sumBDownTime(IN newDowntime:
           REAL);
 END OBJECT;
```

```
    PROCEDURE initializeParams(OUT AparameterQ,
                        BparameterQ:
                        queueListObj);
    PROCEDURE changeQtrlyParams(IN qtr,prevQtr :
                         INTEGER;
                         INOUT AparameterQ,
                         BparameterQ : queueListObj;
                         INOUT update : STRING);
    PROCEDURE UAlambda(IN number, time : INTEGER ;
                  OUT newlambda : REAL);
    PROCEDURE UBlambda(IN number,time : INTEGER;
                  OUT newlambda : REAL);
    PROCEDURE DAlambda(IN number, time : INTEGER ;
                  OUT newlambda : REAL);
    PROCEDURE DBlambda(IN number,time : INTEGER;
                  OUT newlambda : REAL);
    PROCEDURE dailyAUpdate (IN qtr : INTEGER;
                     IN TimeCount : TimeObj;
                     INOUT parameterA :A1000Obj);
    PROCEDURE dailyBUpdate (IN qtr : INTEGER;
                     IN TimeCount : TimeObj;
                     INOUT parameterB :B2000Obj);


END {DEFINITION} MODULE {param}.
```

```
IMPLEMENTATION MODULE param;

{****************************************************************

    MODULE NAME : Iparam          DATE WRITTEN :  13 July 92
    AUTHOR :       B. F. Mimms    LAST MODIFIED :  08 Aug 92
                   CAPT   USMC

    DESCRIPTION : Contains parameter queues, and updating timer object. Contains
                  methods and procedures to reference and update
                  parameters on a daily and quarterly basis.

****************************************************************}

   FROM  queueL         IMPORT queueListObj;
   FROM  global         IMPORT qtrCheck,utauCheck,
                         unumCheck,dtauCheck,
                         dnumCheck, utauHold,
                         unumHold, dtauHold,
                         dnumHold, newlambda;
   FROM Debug           IMPORT TraceStream;

   OBJECT A1000Obj;

      ASK METHOD changeQtr (IN newqtr : INTEGER);
      BEGIN
        qtr := newqtr;
      END METHOD;

      ASK METHOD changeUlambda (IN newulambda  : REAL);
      BEGIN
        ulambda := newulambda;
      END METHOD;

      ASK METHOD changeUtau (IN newutau : INTEGER);
      BEGIN
        utau := newutau;
      END METHOD;

      ASK METHOD changeUnum (IN newunum : INTEGER);
      BEGIN
        unum := newunum;
      END METHOD;
```

```
    ASK METHOD changeDlambda (IN newdlambda : REAL);
    BEGIN
     dlambda := newdlambda;
    END METHOD;

    ASK METHOD changeDtau (IN newdtau : INTEGER);
    BEGIN
      dtau := newdtau;
    END METHOD;

    ASK METHOD changeDnum (IN newdnum : INTEGER);
    BEGIN
      dnum := newdnum;
    END METHOD;

END OBJECT {A1000Obj};

OBJECT B2000Obj;

    ASK METHOD changeQtr (IN newqtr : INTEGER);
    BEGIN
      qtr := newqtr;
    END METHOD;

    ASK METHOD changeUlambda (IN newulambda  : REAL);
    BEGIN
      ulambda := newulambda;
    END METHOD;

    ASK METHOD changeUtau (IN newutau : INTEGER);
    BEGIN
      utau := newutau;
    END METHOD;

    ASK METHOD changeUnum (IN newunum : INTEGER);
    BEGIN
      unum := newunum;
    END METHOD;

    ASK METHOD changeDlambda (IN newdlambda : REAL);
    BEGIN
      dlambda := newdlambda;
    END METHOD;
```

```
    ASK METHOD changeDtau (IN newdtau : INTEGER);
    BEGIN
      dtau := newdtau;
    END METHOD;

    ASK METHOD changeDnum (IN newdnum : INTEGER);
    BEGIN
      dnum := newdnum;
    END METHOD;

END OBJECT {B2000Obj};

OBJECT TimeObj;

ASK METHOD sumAUpTime(IN newUptime : REAL);
VAR
    oldUptime,newIUptime : INTEGER;
BEGIN
    newIUptime := ROUND(newUptime);
    oldUptime := A1000UpTime;
    A1000UpTime := oldUptime + newIUptime;
    INC(A1000CountUp);
END METHOD;

ASK METHOD sumADownTime (IN newDowntime : REAL);
VAR
    oldDowntime, newIDowntime : INTEGER;
BEGIN
    newIDowntime := ROUND(newDowntime);
    oldDowntime := A1000DownTime;
    A1000DownTime := oldDowntime + newIDowntime;
    INC(A1000CountDown);
END METHOD;

ASK METHOD sumBUpTime (IN newUptime : REAL);
VAR
    oldUptime,newIUptime : INTEGER;
BEGIN
    newIUptime := ROUND(newUptime);
    oldUptime := B2000UpTime;
    B2000UpTime := oldUptime + newIUptime;
    INC(B2000CountDown);
END METHOD;
```

```
ASK METHOD sumBDownTime(IN newDowntime :REAL);
VAR
   oldDowntime,newIDowntime : INTEGER;
BEGIN
   newIDowntime := ROUND(newDowntime);
   oldDowntime := B2000DownTime;
   B2000DownTime := oldDowntime + newIDowntime;
   INC(B2000CountDown);
END METHOD;

END OBJECT;


{************************************************************}
   PROCEDURE initializeParams (OUT AparameterQ,
    BparameterQ: queueListObj);
{************************************************************}

   VAR
      qtrA  : A1000Obj;
      qtrB : B2000Obj;

   BEGIN

      NEW (AparameterQ);
      NEW (BparameterQ);

      NEW (qtrA);
      NEW (qtrB);

         ASK qtrA TO changeQtr (1);
         ASK qtrA TO changeUlambda(14.57);
         ASK qtrA TO changeUtau(260);
         ASK qtrA TO changeUnum(20);
         ASK qtrA TO changeDlambda(6.83);
         ASK qtrA TO changeDtau(133);
         ASK qtrA TO changeDnum(19);

         ASK qtrB TO changeQtr(1);
         ASK qtrB TO changeUlambda(26.1);
         ASK qtrB TO changeUtau(253);
         ASK qtrB TO changeUnum(10);
         ASK qtrB TO changeDlambda(3.07);
```

```
            ASK qtrB TO changeDtau(46);
            ASK qtrB TO changeDnum(11);

        ASK AparameterQ TO  Add (qtrA);
        ASK BparameterQ TO Add (qtrB);

        NEW (qtrA);
        NEW (qtrB);
            ASK qtrA TO changeQtr(2);
            ASK qtrB TO changeQtr(2);

        ASK AparameterQ TO Add (qtrA);
        ASK BparameterQ TO Add (qtrB);

        NEW (qtrA);
        NEW (qtrB);
            ASK qtrA TO changeQtr(3);
            ASK qtrB TO changeQtr(3);

        ASK AparameterQ TO Add (qtrA);
        ASK BparameterQ TO Add (qtrB);

        NEW (qtrA);
        NEW (qtrB);
            ASK qtrA TO changeQtr(4);
            ASK qtrB TO changeQtr(4);

        ASK AparameterQ TO Add (qtrA);
        ASK BparameterQ TO Add (qtrB);

    END PROCEDURE {initializeParams};

{***********************************************************}
    PROCEDURE changeQtrlyParams(IN qtr, prevQtr : INTEGER;
        INOUT AparameterQ,BparameterQ:queueListObj;
        INOUT update : STRING);
{***********************************************************}
    VAR
        qtrArec : A1000Obj;
        qtrBrec : B2000Obj;
```

```
BEGIN
   qtrArec := ASK AparameterQ First();
   WHILE qtrArec <> NILOBJ
      qtrCheck    := qtrArec.qtr;
      utauCheck   := qtrArec.utau;
      unumCheck   := qtrArec.unum;
      dtauCheck   := qtrArec.dtau;
      dnumCheck   := qtrArec.dnum;

      IF qtrCheck = prevQtr

         utauHold  := utauCheck;
         unumHold:= unumCheck;

         dtauHold  := dtauCheck;
         dnumHold:= dnumCheck;

       ELSIF qtrCheck = qtr
            ASK qtrArec TO changeUtau(utauHold);
            ASK qtrArec TO changeUnum(unumHold);
            ASK qtrArec TO changeDtau(dtauHold);
            ASK qtrArec TO changeDnum(dnumHold);
            UAlambda(utauHold,unumHold,newlambda);
            ASK qtrArec TO changeUlambda(newlambda);
            DAlambda(dtauHold,dnumHold,newlambda);
            ASK qtrArec TO changeDlambda(newlambda);
            update := "NO";
        END IF;

   qtrArec := ASK AparameterQ Next(qtrArec);
   END WHILE;

   qtrBrec := ASK BparameterQ First();
   WHILE qtrBrec <> NILOBJ

      qtrCheck     := qtrBrec.qtr;
      utauCheck   := qtrBrec.utau;
      unumCheck := qtrBrec.unum;
      dtauCheck   := qtrBrec.dtau;
      dnumCheck := qtrBrec.dnum;
```

```
      IF qtrCheck = prevQtr

         utauHold  := utauCheck;
         unumHold:= unumCheck;

         dtauHold  := dtauCheck;
         dnumHold:= dnumCheck;
       ELSIF qtrCheck = qtr

            ASK qtrBrec TO changeUtau(utauHold);
            ASK qtrBrec TO changeUnum(unumHold);
            ASK qtrBrec TO changeDtau(dtauHold);
            ASK qtrBrec TO changeDnum(dnumHold);
            UBlambda(utauHold,unumHold,newlambda);
            ASK qtrBrec TO changeUlambda(newlambda);
            DBlambda(dtauHold,dnumHold,newlambda);
            ASK qtrBrec TO changeDlambda(newlambda);
            update := "NO"
        END IF;

   qtrBrec := ASK BparameterQ Next(qtrBrec);
   END WHILE;

 END PROCEDURE {changeQtrlyParams};

{***********************************************************}
 PROCEDURE UAlambda(IN time, number : INTEGER;
                OUT newlambda : REAL);
{***********************************************************}

 CONST
    alpha =  0.532;
    beta  =  7.75124;

 BEGIN
    newlambda :=(beta + FLOAT(time)) /(alpha + FLOAT(number)) ;
 END PROCEDURE;
```

```
{************************************************************}
   PROCEDURE DAlambda(IN time, number : INTEGER;
                  OUT newlambda : REAL);
{************************************************************}

   CONST
      alpha = 0.93;
      beta  = 6.3519;

   BEGIN
      newlambda := (beta + FLOAT(time)) / (alpha + FLOAT(number)) ;
   END PROCEDURE;

{************************************************************}
   PROCEDURE UBlambda(IN time,number : INTEGER;
                  OUT newlambda : REAL);
{************************************************************}

   CONST
      alpha = 0.7931;
      beta  = 20.7;

   BEGIN
      newlambda := (beta + FLOAT(time)) / (alpha + FLOAT(number));
   END PROCEDURE;

{************************************************************}
   PROCEDURE DBlambda(IN time,number : INTEGER;
                  OUT newlambda : REAL);
{************************************************************}

   CONST
      alpha   = 0.85;
      beta    = 2.6095;

   BEGIN
      newlambda := (beta + FLOAT(time))  / (alpha + FLOAT(number));
   END PROCEDURE;
```

```
{**********************************************************}
   PROCEDURE dailyAUpdate (IN quarter : INTEGER;
                           IN TimeCount : TimeObj;
                           INOUT parameterA : A1000Obj) ;
{**********************************************************}

   VAR
      newAutau, newAunum, newAdtau , newAdnum : INTEGER;
   BEGIN
    newAutau  := parameterA.utau + TimeCount.A1000UpTime;
    ASK parameterA TO changeUtau(newAutau);
    newAunum  := parameterA.unum + TimeCount.A1000CountUp;
    ASK parameterA TO changeUnum(newAunum);
    newAdtau  := parameterA.dtau + TimeCount.A1000DownTime;
    ASK parameterA TO changeDtau(newAdtau);
    newAdnum  := parameterA.dnum + TimeCount.A1000CountDown;
    ASK parameterA TO changeDnum(newAdnum);
   END PROCEDURE;

{**********************************************************}
   PROCEDURE dailyBUpdate (IN qtr : INTEGER;
                           IN TimeCount : TimeObj;
                           INOUT parameterB : B2000Obj);
{**********************************************************}

   VAR
      newButau, newBunum, newBdtau, newBdnum : INTEGER;
   BEGIN
    newButau  := parameterB.utau + TimeCount.B2000UpTime;
    ASK parameterB TO changeUtau(newButau);
    newBunum:= parameterB.unum + TimeCount.B2000CountUp;
    ASK parameterB TO changeUnum(newBunum);
    newBdtau  := parameterB.dtau + TimeCount.B2000DownTime;
    ASK parameterB TO changeDtau(newBdtau);
    newBdnum:= parameterB.dnum + TimeCount.B2000CountDown;
    ASK parameterB TO changeDnum(newBdnum);
   END PROCEDURE;

END {IMPLEMENTATION} MODULE {param}.
```

```
DEFINITION MODULE predict;

{*****************************************************************

     MODULE NAME : Dpredict        DATE WRITTEN :  15 July 92
     AUTHOR :      B. F. Mimms     LAST MODIFIED : 19 July 92
                   CAPT   USMC

     DESCRIPTION : Contains procedures to read seeds from input file, access those
               seeds, and provide UP and DOWN  predictions

*****************************************************************}

    FROM RandMod         IMPORT RandomObj;
    FROM endItem         IMPORT endItemObj;
    FROM queueL          IMPORT queueListObj;
    FROM IOMod           IMPORT ALL  FileUseType, StreamObj;
    FROM param           IMPORT A1000Obj, B2000Obj;
    FROM Debug           IMPORT TraceStream;

    PROCEDURE predictUp (INOUT n,qtr : INTEGER;
                   IN AparameterQ, BparameterQ :
                   queueListObj; IN idno : STRING;
                   OUT pred : REAL);
    PROCEDURE predictDown (INOUT n, qtr : INTEGER;
                    IN AparameterQ, BparameterQ:
                    queueListObj;IN idno : STRING;
                    OUT pred : REAL);
    PROCEDURE grabSeed(INOUT n : INTEGER; OUT seed :
                  INTEGER);
    PROCEDURE readSeeds();

END {DEFINITION} MODULE {predict}.
```

```
IMPLEMENTATION MODULE predict;

{************************************************************
     MODULE NAME : Ipredict     DATE WRITTEN :  15 July 92
     AUTHOR :       B. F. Mimms   LAST MODIFIED : 19 July 92
                    CAPT   USMC

     DESCRIPTION : Contains procedures to read seeds from input file, access those
               seeds, and provide UP and DOWN predictions

************************************************************}

   FROM RandMod        IMPORT RandomObj;
   FROM endItem        IMPORT endItemObj;
   FROM queueL         IMPORT queueListObj;
   FROM param          IMPORT TimeObj, A1000Obj, B2000Obj;
   FROM IOMod          IMPORT  ALL FileUseType, StreamObj;
   FROM global         IMPORT seedArray;
   FROM Debug          IMPORT TraceStream;

{************************************************************}
   PROCEDURE predictUp (INOUT n, qtr : INTEGER;
                  IN AparameterQ, BparameterQ :
                  queueListObj;IN idno : STRING;
                  OUT pred : REAL);
{************************************************************}

   VAR
      streamUp, streamDown : RandomObj;
      Aestimate            : A1000Obj;
      Bestimate            : B2000Obj;
      alambda, blambda     : REAL;
      seed                 : INTEGER;

   BEGIN
      NEW (streamUp);
      grabSeed(n,seed);

      ASK streamUp TO SetSeed (seed);
      IF idno = "A1000"
      Aestimate := ASK AparameterQ First();
      WHILE Aestimate <> NILOBJ
```

```
        IF qtr = Aestimate.qtr
          alambda := Aestimate.ulambda;
          pred := streamUp.Exponential(alambda);
        END IF;
     Aestimate := ASK AparameterQ Next(Aestimate);
     END WHILE;
     END IF;

     IF idno = "B2000"
     Bestimate := ASK BparameterQ First();
     WHILE Bestimate <> NILOBJ
        IF qtr = Bestimate.qtr
          blambda := Bestimate.ulambda;
          pred := streamUp.Exponential(blambda);
        END IF;
      Bestimate := ASK BparameterQ Next(Bestimate);
      END WHILE;
      END IF;
      DISPOSE(streamUp);

  END PROCEDURE;

{*******************************************************}
  PROCEDURE predictDown (INOUT n, qtr : INTEGER;
                    IN AparameterQ, BparameterQ :
                    queueListObj;IN idno : STRING;
                    OUT pred : REAL);
{*******************************************************}

  VAR
     streamDown       : RandomObj;
     Aestimate        : A1000Obj;
     Bestimate        : B2000Obj;
     alambda,blambda  : REAL;
     seed             : INTEGER;

  BEGIN
     NEW (streamDown);
     grabSeed(n,seed);

     ASK streamDown TO SetSeed (seed);
     IF idno = "A1000"
     Aestimate := ASK AparameterQ First();
```

```
      WHILE Aestimate < > NILOBJ
         IF qtr = Aestimate.qtr
            alambda := Aestimate.dlambda;
            pred := streamDown.Exponential(alambda);
         END IF;
      Aestimate := ASK AparameterQ Next(Aestimate);
      END WHILE;
      END IF;

      IF idno = "B2000"
      Bestimate := ASK BparameterQ First();
      WHILE Bestimate < > NILOBJ
         IF qtr = Bestimate.qtr
            blambda := Bestimate.dlambda;
            pred := streamDown.Exponential(blambda);
         END IF;
       Bestimate := ASK BparameterQ Next(Bestimate);
       END WHILE;
       END IF;
       DISPOSE(streamDown);

   END PROCEDURE;

{***************************}
   PROCEDURE readSeeds ();
{***************************}

   VAR
      strmIn  : StreamObj;
      str     : STRING;
      i       : INTEGER;
      NumSeeds: INTEGER;
   BEGIN
      NEW (strmIn);
      ASK strmIn TO Open("seeds.dat", Input);
      ASK strmIn TO ReadInt (NumSeeds);
      NEW (seedArray, 1..NumSeeds);

      FOR i := 1 TO NumSeeds
          ASK strmIn TO ReadInt(seedArray[i]);
          ASK strmIn TO ReadLine(str);
      END FOR;
   END PROCEDURE;
```

```
{*************************************************************}
    PROCEDURE grabSeed (INOUT n : INTEGER; OUT seed :
     INTEGER);
{*************************************************************}

    BEGIN
        seed := seedArray[n];
    END PROCEDURE;

END {IMPLEMENTATION} MODULE {predict}.
```

```
DEFINITION MODULE upDater;

{*************************************************************
     MODULE NAME : Dupdater      DATE WRITTEN :  13 July 92
     AUTHOR :       B. F. Mimms   LAST MODIFIED : 13 July 92
                    CAPT   USMC

     DESCRIPTION : Contains procedure that identifies correct quarter,
              and determines if a quarterly update is necessary

*************************************************************}

      FROM global          IMPORT  todaysDate,qDiff,
                                    update,done1, done2,
                                    done3, done4,firstTime, qtr,prevQtr;
      FROM param           IMPORT  A1000Obj, B2000Obj;
      FROM queueL          IMPORT  queueListObj;
      FROM Debug           IMPORT  TraceStream;

      PROCEDURE checkUpdate (INOUT todaysDate  : REAL;
                       OUT qtr: INTEGER;
                       OUT AparameterQ, BparameterQ:
                       queueListObj;
                       INOUT update,done1,done2,
                       done3,done4,firstTime:
                       STRING);

END {DEFINITION} MODULE {upDater}.
```

```
IMPLEMENTATION MODULE upDater;

{**************************************************************

    MODULE NAME : Iupdater       DATE WRITTEN :  13 July 92
    AUTHOR :       B. F. Mimms    LAST MODIFIED : 13 July 92
                   CAPT   USMC

    DESCRIPTION : Contains procedure that identifies correct quarter, and
                  determines if a quarterly update is necessary

**************************************************************}

     FROM global   IMPORT  todaysDate, qDiff, update,
                    done1, done2, done3, done4,
                    firstTime, qtr, prevQtr,
                    jDiff;
     FROM proced   IMPORT  julianDiff;
     FROM param    IMPORT  A1000Obj, B2000Obj,
                    changeQtrlyParams;
     FROM queueL   IMPORT  queueListObj;
     FROM Debug    IMPORT  TraceStream;

{**************************************************************}
     PROCEDURE checkUpdate (INOUT todaysDate : REAL;
                    OUT qtr : INTEGER;
                    OUT  AparameterQ, BparameterQ:
                    queueListObj;
                    INOUT update,done1,done2,done3,
                    done4,firstTime: STRING);
{**************************************************************}

     VAR
        baseVal, baseDate     : REAL;

     CONST
        thou  = 1000.0;

     BEGIN
        baseVal  := FLOAT (ROUND (todaysDate/thou));
        baseDate := baseVal * thou;
        julianDiff (baseDate, todaysDate, jDiff);
        qDiff := TRUNC (jDiff);
```

```
CASE qDiff
     WHEN 270..366:
          qtr:=4;
          prevQtr := 3;
          IF done4 = "NO"
             update := "YES";
             done4 := "YES";
          END IF;
     WHEN 180..269:
          qtr :=3;
          prevQtr := 2;
          IF done3 = "NO"
             update := "YES";
             done3 := "YES";
          END IF;
     WHEN 90..179:
          qtr:=2;
          prevQtr := 1;
          IF done2 = "NO"
             update := "YES";
             done2 := "YES";
             update := "YES";
          END IF;
     OTHERWISE
          qtr:=1;
          IF firstTime = "NO"
             IF done1 = "NO"
                update := "YES";
                done1 := "YES";
             END IF;
          ELSE
             firstTime := "NO";
          END IF;
END CASE;

IF update = "YES"
     changeQtrlyParams(qtr, prevQtr,AparameterQ,BparameterQ,
                       update);
END IF;

END PROCEDURE {checkUpdate};

END {IMPLEMENTATION} MODULE {upDater}.
```

```
DEFINITION MODULE simulat;

{***********************************************************}

     MODULE NAME : Dsimulat     DATE WRITTEN :   20 July 92
     AUTHOR :      B. F. Mimms  LAST MODIFIED :  28 July 92
                   CAPT   USMC

     DESCRIPTION : Contains techniques for setting up suite
                   simulation, TELL methods for commencing
                   simulation, and statistics collecting variables

***********************************************************}

    FROM predict    IMPORT grabSeed;
    FROM RandMod    IMPORT RandomObj;
    FROM queueL     IMPORT queueListObj;
    FROM StatMod    IMPORT RStatObj;
    FROM endItem    IMPORT endItemObj;
    FROM Debug      IMPORT TraceStream;

    TYPE
      simEndItemObj = OBJECT
          idno     : STRING;
          deadline : STRING;
          ASK METHOD changeid (IN newid : STRING);
          ASK METHOD changedl (IN newdl : STRING);
          TELL METHOD skedUp(IN phaseTime,quitTime,
                         upLambda,downLambda :
                         REAL;IN simStreamUp,
                         simStreamDown: RandomObj;
                         IN idnum : STRING);
          TELL METHOD skedDown(IN phaseTime,quitTime,
                          upLambda,downLambda :
                          REAL;IN simStreamUp,
                          simStreamDown:
                          RandomObj;IN idnum :
                          STRING);
      END OBJECT;
```

```
PROCEDURE commenceSim(IN alphaUpLambda,
                         alphaDownLambda,
                         betaUpLambda,
                         betaDownLambda,
                         quitTime : REAL;
                         IN simListing :
                         queueListObj);

PROCEDURE resetStats;

VAR
    Auptime        : LMONITORED REAL BY RStatObj;
    Buptime        : LMONITORED REAL BY RStatObj;
    Adowntime      : LMONITORED REAL BY RStatObj;
    Bdowntime      : LMONITORED REAL BY RStatObj;
    AuptimeStats   : RStatObj;
    BuptimeStats   : RStatObj;
    AdowntimeStats : RStatObj;
    BdowntimeStats : RStatObj;
    simListing     : queueListObj;
    simStreamUpA1,simStreamUpA2,simStreamUpA3,
    simStreamUpB1,simStreamUpB2,simStreamUpB3,
    simStreamDownA1,simStreamDownA2,simStreamDownA3,
    simStreamDownB1,simStreamDownB2,simStreamDownB3,
    simStreamUpA,simStreamUpB,simStreamDownA,
    simStreamDownB  : RandomObj;
    seedUpA1,seedUpA2,seedUpA3,
    seedUpB1,seedUpB2,seedUpB3,
    seedDownA1,seedDownA2,seedDownA3,
    seedDownB1,seedDownB2,seedDownB3 : INTEGER;
    member : endItemObj;
    newid,newdl,newsn,idnum,dl,sernum: STRING;
    phaseTime : REAL;
    simMember : simEndItemObj;
    seedUpA,
    seedUpB,
    seedDownA,
    seedDownB      : INTEGER;
    member         : endItemObj;
    newid,newdl,
    idnum,dl       : STRING;
    phaseTime      : REAL;
    simMember      : simEndItemObj;
```

```
    PROCEDURE initializeSuiteSimulation (INOUT n : INTEGER;
              IN listing :  queueListObj;
              OUT simListing : queueListObj);

END {DEFINITION} MODULE {simulat}.


IMPLEMENTATION MODULE simulat;

{*************************************************************
    MODULE NAME : Isimulat        DATE WRITTEN :   20 July 92
    AUTHOR :      B. F. Mimms     LAST MODIFIED :  25 Aug  92
                  CAPT   USMC

    DESCRIPTION : Contains techniques for setting up suite
              simulation, TELL methods for commencing
              simulation, and statistics collecting variables

*************************************************************

    FROM StatMod   IMPORT RStatObj;
    FROM RandMod   IMPORT RandomObj;
    FROM predict   IMPORT grabSeed;
    FROM queueL    IMPORT queueListObj;
    FROM endItem   IMPORT endItemObj;
    FROM SimMod    IMPORT SimTime,StartSimulation;
    FROM Debug     IMPORT TraceStream;

{*************************************************************}
    PROCEDURE initializeSuiteSimulation (INOUT n : INTEGER;
                   IN listing : queueListObj;
                   OUT simListing : queueListObj);
{*************************************************************}

    BEGIN

    NEW (AuptimeStats);
    ADDMONITOR (Auptime,AuptimeStats);

    NEW(AdowntimeStats);
    ADDMONITOR (Adowntime, AdowntimeStats);
```

```
NEW (BuptimeStats);
ADDMONITOR (Buptime, BuptimeStats);

NEW (BdowntimeStats);
ADDMONITOR (Bdowntime, BdowntimeStats);

NEW (simStreamUpA1);
NEW (simStreamDownA1);
NEW (simStreamUpA2);
NEW (simStreamDownA2);
NEW (simStreamUpA3);
NEW (simStreamDownA3);
NEW (simStreamUpB1);
NEW (simStreamDownB1);
NEW (simStreamUpB2);
NEW (simStreamDownB2);
NEW (simStreamUpB3);
NEW (simStreamDownB3);

grabSeed (n,seedUpA1);
INC (n);
grabSeed (n,seedUpA2);
INC (n);
grabSeed (n,seedUpA3);
INC (n);
grabSeed (n,seedUpB1);
INC (n);
grabSeed (n,seedUpB2);
INC (n);
grabSeed (n,seedUpB3);
INC (n);
grabSeed (n,seedDownA1);
INC (n);
grabSeed (n,seedDownA2);
INC (n);
grabSeed (n,seedDownA3);
INC (n);
grabSeed (n,seedDownB1);
INC (n);
grabSeed (n,seedDownB2);
INC (n);
grabSeed (n,seedDownB3);
INC (n);
```

```
ASK simStreamUpA1 TO SetSeed (seedUpA1);
ASK simStreamUpA2 TO SetSeed (seedUpA2);
ASK simStreamUpA3 TO SetSeed (seedUpA3);
ASK simStreamDownA1 TO SetSeed (seedDownA1);
ASK simStreamDownA2 TO SetSeed (seedDownA2);
ASK simStreamDownA3 TO SetSeed (seedDownA3);
ASK simStreamUpB1 TO SetSeed (seedUpB1);
ASK simStreamUpB2 TO SetSeed (seedUpB2);
ASK simStreamUpB3 TO SetSeed (seedUpB3);
ASK simStreamDownB1 TO SetSeed (seedDownB1);
ASK simStreamDownB2 TO SetSeed (seedDownB2);
ASK simStreamDownB3 TO SetSeed (seedDownB3);

NEW(simListing);
member := ASK listing First();
WHILE member <> NILOBJ

    newid := member.idno;
    newdl := member.deadline;
    newsn:= member.sernum;
    NEW (simMember);
        ASK simMember TO changeid(newid);
        ASK simMember TO changedl(newdl);
        ASK simMember TO changesn(newsn);
        ASK simListing TO Add (simMember);
member := ASK listing Next(member);
END WHILE;

END PROCEDURE;


OBJECT simEndItemObj;
    ASK METHOD changeid (IN newid : STRING);
    BEGIN
        idno := newid;
    END METHOD;

    ASK METHOD changedl (IN newdl : STRING);
    BEGIN
        deadline := newdl;
    END METHOD;
```

```
ASK METHOD changesn (IN newsn : STRING);
BEGIN
     sernum := newsn;
END METHOD;

TELL METHOD skedUp(IN phaseTime,quitTime,
                upLambda,downLambda : REAL;
                IN simStreamUp,simStreamDown:
                RandomObj; IN idnum : STRING);
BEGIN
    IF SimTime() > quitTime TERMINATE;
    END IF;
    IF idnum = "A1000"
        Adowntime := SimTime() - phaseTime;
    ELSE
        Bdowntime := SimTime() - phaseTime;
    END IF;
    phaseTime := SimTime();
    TELL SELF TO skedDown(phaseTime,quitTime,
                    upLambda,downLambda,
                    simStreamUp,
                    simStreamDown,idnum)
                    IN simStreamUp.
                    Exponential(upLambda);
END METHOD;

TELL METHOD skedDown(IN phaseTime,quitTime,
                upLambda,downLambda : REAL;
                IN simStreamUp,simStreamDown:
                RandomObj; IN idnum : STRING);

BEGIN
    IF SimTime() > quitTime TERMINATE;
    END IF;
    IF idnum = "A1000"
        Auptime := SimTime() - phaseTime;
    ELSE
        Buptime := SimTime() - phaseTime;
    END IF;
    phaseTime := SimTime();
    TELL SELF TO skedUp(phaseTime,quitTime,
                    upLambda,downLambda,
                    simStreamUp,
```

```
                        simStreamDown,idnum)
                        IN simStreamDown.
                        Exponential(downLambda);
   END METHOD;
END OBJECT;

PROCEDURE commenceSim( IN alphaUpLambda,
               alphaDownLambda, betaUpLambda,
               betaDownLambda, quitTime :
               REAL; IN simListing : queueListObj);
VAR
  simMember  : simEndItemObj;
BEGIN
  simMember := ASK simListing First();
  WHILE simMember <> NILOBJ
       idnum := simMember.idno;
       dl := simMember.deadline;
       sernum := simMember.sernum;
       IF idnum = "A1000"
         IF sernum = "A1"
           simStreamDownA := simStreamDownA1;
           simStreamUpA := simStreamUpA1;
         ELSIF sernum = "A2"
           simStreamDownA := simStreamDownA2;
           simStreamUpA := simStreamUpA2;
         ELSE
           simStreamDownA := simStreamDownA3;
           simStreamUpA := simStreamUpA3;
         END IF;

         IF dl = "NO"
           phaseTime := SimTime();
           TELL simMember TO skedDown(phaseTime,
                     quitTime,alphaUpLambda,
                     alphaDownLambda,
                     simStreamUpA,
                     simStreamDownA,idnum)
                     IN simStreamUpA.Exponential(alphaUpLambda);
         ELSE {deadline = "YES"}
           TELL simMember TO skedUp(phaseTime,
                     quitTime,alphaUpLambda,
                     alphaDownLambda,
                     simStreamUpA,
```

```
                        simStreamDownA,idnum)
                        IN simStreamDownA.Exponential(alphaDownLambda);
            END IF;
          ELSE {idno = "B2000"}
            IF sernum = "B1"
              simStreamDownB := simStreamDownB1;
              simStreamUpB := simStreamUpB1;
            ELSIF sernum = "B2"
              simStreamDownB := simStreamDownB2;
              simStreamUpB := simStreamUpB2;
            ELSE
              simStreamDownB := simStreamDownB3;
              simStreamUpB := simStreamUpB3;
            END IF;
            IF dl = "NO"
              phaseTime := SimTime();
              TELL simMember TO skedDown(phaseTime,
                        quitTime,betaUpLambda,
                        betaDownLambda,
                        simStreamUpB,
                        simStreamDownB,idnum)
                        IN simStreamUpB.Exponential(betaUpLambda);
            ELSE {deadline = "YES"}
              TELL simMember TO skedUp(phaseTime,
                        quitTime,betaUpLambda,
                        betaDownLambda,
                        simStreamUpB,
                        simStreamDownB,idnum)
                        IN simStreamDownB.Exponential(betaDownLambda);
            END IF;
          END IF;
      simMember := ASK simListing Next(simMember);
      END WHILE;
  END PROCEDURE;

  PROCEDURE resetStats;
  BEGIN
     Auptime := 0.0;
     Adowntime := 0.0;
     Buptime := 0.0;
     Bdowntime := 0.0;
  END PROCEDURE;
END {IMPLEMENTATION} MODULE {simulat}.
```

```
DEFINITION MODULE output;

{***********************************************************}

     MODULE NAME : Doutput       DATE WRITTEN :  12 July 92
     AUTHOR :      B. F. Mimms   LAST MODIFIED : 08 Aug  92
                   CAPT   USMC

     DESCRIPTION : Contains output routines for the model.

***********************************************************}

   FROM queueL        IMPORT queueListObj;
   FROM endItem       IMPORT endItemObj;
   FROM global        IMPORT listing, roster;
   FROM Debug         IMPORT TraceStream;

   PROCEDURE simOutput (IN todaysDate,quitDate,quitTime :
                 REAL; IN day : INTEGER);

END {DEFINITION} MODULE {output}.


IMPLEMENTATION MODULE output;

{***********************************************************}

     MODULE NAME : Ioutput       DATE WRITTEN :  12 July 92
     AUTHOR :      B. F. Mimms   LAST MODIFIED : 08 Aug 92
                   CAPT   USMC

     DESCRIPTION : Contains output routines for the model.

***********************************************************}

   FROM endItem        IMPORT endItemObj;
   FROM transac        IMPORT transactionObj;
   FROM queueL         IMPORT queueListObj;
   FROM global         IMPORT listing,member, roster,
                        transact, todaysDate;
   FROM Debug          IMPORT TraceStream;
   FROM simulat        IMPORT AuptimeStats,
```

```
                        AdowntimeStats,
                        BuptimeStats,
                        BdowntimeStats;
FROM IOMod              IMPORT StreamObj, FileUseType(Output);

VAR
    Strm                  : StreamObj;
    Aup,Adown,Bup,Bdown,Run,A,B : STRING;
CONST

     headSim    =  " Suite Simulation    ";
     headSim1   =  "  FROM   TO      FOR (days)  ";
     formatSim1 =  "  *****     *****       ***  ";
     headSim2   =  "Results :";
     headSim3="Item ID # Operational Availability(%)";
     formatSim2=  "*****        **.*** ";
     headSim4   =  "Aup      Adown  Bup    Bdown";
     formatSim3=  "**.**    **.**   **.**   **.**";

{********************************************************}
   PROCEDURE simOutput(IN todaysDate,quitDate,quitTime :
REAL; IN day : INTEGER);
{********************************************************}

   VAR
      AoA, AoB  : REAL;
   BEGIN
      AoA := 100.0 * (AuptimeStats.Mean() /(AuptimeStats.Mean() +
           AdowntimeStats.Mean()));
      AoB := 100.0 * BuptimeStats.Mean() / (BuptimeStats.Mean() +
           BdowntimeStats.Mean());
      A   := "A1000";
      B   := "B2000";

      NEW(Strm);

      IF day = 1
         ASK Strm TO Open ("sim1out.txt",Output);
      ELSIF day = 2
         ASK Strm TO Open ("sim2out.txt",Output);
      ELSIF day = 3
         ASK Strm TO Open ("sim3out.txt",Output);
      END IF;
```

```
      ASK Strm TO WriteString(headSim);
      ASK Strm TO WriteLn;
      ASK Strm TO WriteString(headSim1);
      ASK Strm TO WriteLn;
      ASK Strm TO WriteString(SPRINT(todaysDate,quitDate,quitTime)
              WITH formatSim1);
      ASK Strm TO WriteLn;
      ASK Strm TO WriteString(headSim2);
      ASK Strm TO WriteLn;
      ASK Strm TO WriteString(headSim3);
      ASK Strm TO WriteLn;
      ASK Strm TO WriteString(SPRINT(A,AoA) WITH formatSim2);
      ASK Strm TO WriteLn;
      ASK Strm TO WriteString(SPRINT(B,AoB) WITH formatSim2);
      ASK Strm TO WriteLn;

      ASK Strm TO Close;
      DISPOSE(Strm);

   END PROCEDURE;

END {IMPLEMENTATION} MODULE {output}.
```

LIST OF REFERENCES

1. Marine Corps Order P4790.2b MIMMS Field Procedures Manual, 24 June 1983.

2. Navy Times, 18 May 1992.

3. MAGTF II (Warplanning)/LOG AIS Planning/Deployment and Employment Support System Pamphlet.

4. HQMC, LPS-4, "Development of a Readiness Module for LFADS," Position Paper, 22 Oct 1991.

5. CMC Washington, D.C., "Landing Force Asset Distribution System (LFADS)," Message, 230111Z AUG 91.

6. Operational Handbook (OH) 2, The Marine Air-Ground Task Force, March, 1987.

7. G-3 Operations Officer, 2d Force Service Support Group, "Enhancements To LFADS," Memorandum, 31 May 1991.

8. Marine Corps Users Manual, UM-4790-5, June 1987.

9. Technical Manual, TM-4700-15/1f, Equipment Record Procedures, August, 1988.

10. Ross, Sheldon M., Introduction to Probability Models, Academic Press, Inc., 1989.

11. Esary, J. D., "Operational Availability - MOE or BOF?," NPS, May, 1989.

12. Edwards, John E., MAJ, USA, Combat Service Support Guide, Stackpole Books, 1989.

13. Woods, W. M., "Bayesian Reliability Estimation Methods".

14. DeGroot, M. H., Probability and Statistics, Addison-Wesley, 1975.

15. Gaver, D. P., and Lehoczky, J. P., "Statistical Analysis of Hierarchical Stochastic Models : Examples and Approaches", Annals of Operations Research, v.8, 1987.

16. Executive Director for Logistics Operations, MCLB, Albany, Georgia, "MIMMS Data Extraction," Letter, 2 March, 1992.

INITIAL DISTRIBUTION LIST

		copies
1.	Commandant of the Marine Corps Code TE06 Headquarters, U.S. Marine Corps Washington, D.C. 20380-0001	1
2.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
3.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
4.	Professor Lyn R. Whitaker Code OR/WH Naval Postgraduate School Monterey, CA 93943-5000	4
5.	Professor W. Max Woods OR/WO Naval Postgraduate School Monterey, CA 93943-5000	1
6.	LtCol. J. C. Crutchfield Code LPS-4 Headquarters, U.S. Marine Corps Washington, D.C. 20380-0001	1
7.	LtCol. J. A. O'Donovan, OIC 1st Field Supply and Maintenance Analysis Office (FSMAO) Camp LeJeune, NC 28542-5000	1
8.	Capt. Bernard F. Mimms, Jr. Code LPM-3 Headquarters, U.S. Marine Corps Washhington, D.C. 20380-0001	2

9. Capt. R. Webbers 1
Code 804
Executive Director For Logistics Operations
Marine Corps Logistics Bases
Albany, GA 31704-5000